PL-TR-93-2219

AD-A277 469

(\l

High Frequency Active Auroral Research Program (HAARP) IMAGER

Cyril Lance
Robert H. Eather

Keo Consultants
27 Irving St.
Brookline MA 02146

DTIC
SELECTE
MAR 17, 1994
S B D

30 September, 1993

Final Report
19 August, 1991 - 29 August., 1993

94-08551

PHILLIPS LABORATORY
Directorate of Geophysics
AIR FORCE MATERIEL COMMAND
HANSCOM AIR FORCE BASE, MA 01731-3010

"This technical report has been reviewed and is approved for publication"

EDWARD J. WEBER
Contract Manager

EDWARD J. BERGHORN
Branch Chief

WILLIAM K. VICKERY
Division Director

This document has been reviewed by the ESC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

Qualified requestors may obtain additional copies from the Defense Technical Information Center (DTIC). All others should apply to the National Technical Information Service (NTIS).

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify PL/TSI, 29 Randolph Road, Hanscom AFB, MA 01731-3010. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 30SEP93	3. REPORT TYPE AND DATES COVERED Final 29AUG91 – 29AUG93

4. TITLE AND SUBTITLE

High Frequency Active Auroral Research Program (HAARP) Imager

5. FUNDING NUMBERS

PE61107D
PR 4029 TAO1 WU AC

6. AUTHOR(S)

Cyril Lance

Robert Eather

Contract:
F19628-91-C-0141

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Keo Consultants
27 Irving St.
Brookline   MA   02146

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Phillips Laboratory
29 Randolph Rd.
Hanscom AFB   MA   01731-3010
Contract Manager: E. Weber/GPIA

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

PL-TR-93-2219

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A low-light-level monochromatic imaging system was designed and fabricated, which was optimized to detect and record optical emissions associated with high-power rf heating of the ionosphere. The instrument is capable of detecting very low intensities, of the order of 1 Rayleigh, from typical ionospheric atomic and molecular emissions. This is achieved through co-adding of ON images during heater pulses, and subtraction of OFF (background) images between pulses. Images can be displayed and analysed in real time, and stored on optical disc for later analysis. Full image processing software is provided, which was customized for this application and uses menu or mouse user interaction.

14. SUBJECT TERMS

Low-light-level imager; monochromatic imager;
aurora; airglow; heating experiments

15. NUMBER OF PAGES
322

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

HAARP Imager Documentation

Table of Contents

Chatper 1: Optics Design

Chapter 2: Calibration and Specifications

Chapter 3: Imager Hardware

Chapter 4: Image Processing System

Chapter 5: Using the MIPCTL Software

Chapter 6: Programmer's Details

Chapter 7: FORTH and DSP Programming

Chapter 8: FORTH Code Listings

Chapter 9: KEO Consultants Hardware Schematics

Chapter 10: Advanced Technologies Hardware Schematics

Chapter 11: Cabling and Internal Wiring

Figures

Chapter 1 HAARP Optics Design

1.1 Overview:

The objectives in the design of the HAARP optics were as follows:

(i)   Variable field of view, from $180°$ to ~ $10°$, with telecentric optics so as to allow the use of narrow-band interference filters (~ 2.0 nm bandwidth).

(ii)   Interchangeable filters (5-position filter wheel).

(iv)   Maximum sensitivity and resolution.

(v)   Minimum vignetting.

Normal camera lenses have large ray angles to the principal ray throughout their optical path, and so are unsuitable for use with narrow-band interference filters (see **Figure 1.1**). The shift of transmission wavelength with filter angle increases as the angle squared, so that larger ray angles rapidly requires much wider filters. To overcome this problem, telecentric systems can be designed, where the principal ray of all image-forming cones across the field of view cross the image plane parallel to the optical axis. Thus the maximum ray angles through the filter are determined only by the F number of the lens.

For low-light-level imaging, it is desirable to operate at the lowest F numbers possible. Lower F numbers mean higher ray angles through the filter, and so require wider-band filters. However narrow-band filters are readily available at much larger diameters than the imaging detectors, so that high F number images at the filter can be re-imaged to low F number images at the detector. [The Lagrange Invariant is conserved: $r.\sin \alpha$ = constant, where r is the image size and α is the half-angle of the image-forming cone.]

Thus the philosophy of the HAARP optical system was to use the largest diameter filters readily available ($\leq 4''$) at the desired bandwidths (~ 2.0 nm) to fill the filter area with a higher F number telecentric image, and then to re-image at low F number onto the detector. In this way we can effectively form monochromatic images of wide-angle fields (up to $180°$ fish-eye) onto a 25 mm image intensifier cathode, at an effective F numbers as low as F1.2.

1.2 Fisheye Lens Considerations:

Much of the operation of the HAARP instrument will be with a fisheye lens (180° field of view). The following describes properties of various configurations of such lenses:

A lens that covers a hemispherical field of view (180 degrees) is usually called a fish-eye lens. These lenses have inherent large distortion because it is not possible to form an image of a hemispheric field onto a plane without distortion. This distortion should not be considered as an aberration, but a necessary result of the projection.

There are five projection systems that have been used in fish-eye lens design. If the angle of an incident ray from an infinite object is ϕ and the coordinates of the image point be $r(\phi)$, then the projections are as follows (f is lens focal length):

1. $r \quad = \quad f . \tan \phi$
2. $r \quad = \quad 2f . \tan (\phi/2)$
3. $r \quad = \quad f . \phi$
4. $r \quad = \quad f . \sin \phi$
5. $r \quad = \quad 2f . \sin (\phi/2)$

Projection 1 is that of a normal camera lens.

Projection 2 is called stereographic projection. A small circle on the hemisphere with its center at the lens is projected as a circle on the image plane, but the diameter of a circle at the edge of the image ($\phi = 180^\circ$) is 2x as large as the image of an equally large circle at the pole ($\phi = 0^\circ$). This projection is similar to our psychological perception of the whole sky.

Projection 3 is called equidistant projection, and is preferred for measurements of zenith and azimuth angles because of the linear relation between r and ϕ. However the image of a small circle is not a circle, and is approximately elliptical with the major axis aligned azimuthally.

Projection 4 is called orthographic projection. The area of an image is proportional to illumination by the object on a plane parallel to the film surface.

Projection 5 is called equisolid projection, and the solid angle $d\Omega$ is proportional to the corresponding area of the image. This projection is preferred for measuring relative areas of the sky (for example, % sky covered by clouds).

Almost all commercially available fish-eye lenses for 35 mm and medium format cameras are of the equidistant projection type, which minimizes contraction of the image near the edges of the field.
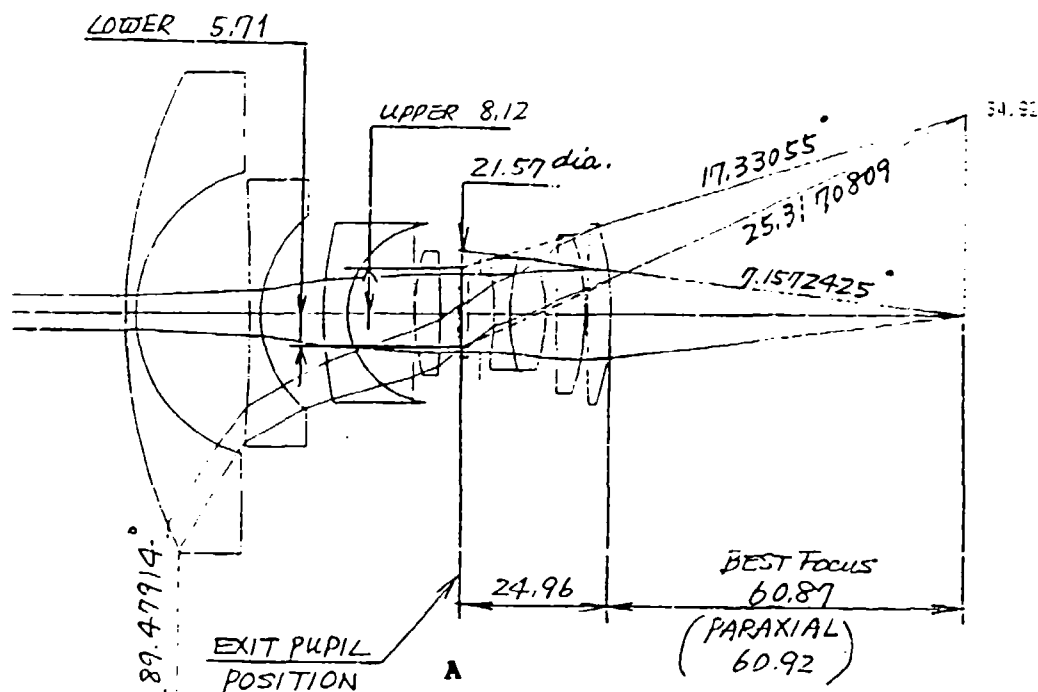
Figure 1.1: Shows a ray diagram for a typical fish-eye lens with F number equal to 4.0 The cone angle at the center of the image is ±7.1° about the principal ray. At the edge of the field of view (image diameter = 70 mm), the cone angle is only ±4.0° about the principal ray; the decrease from ±7.1° is indicative of the vignetting of the lens at large field angles. At the edge of the field, the image-forming cone spans a range of angles from 17.3° - 25.3° to the principal axis.

Clearly it is not possible to use a narrow-band interference filter in this situation, as transmission would vary significantly across the image.

Note: The shift in wavelength of peak transmission for parallel light incident on a filter at angle θ (radians) is given by:

$$\frac{\Delta\lambda}{\lambda} = -\frac{\theta^2}{2\mu^{*2}}$$

where μ* is the effective refractive index.

Similarly, the shift in peak transmission for a cone of light of semi-angle α (radians) incident on a filter with the cone axis at normal incidence is given by:

$$\frac{\Delta\lambda}{\lambda} = -\frac{1}{2} \cdot \frac{\alpha^2}{2\mu^{*2}}$$

3

1.3 Commercial Lens Types:

There are a number of commercial "medium-format" camera lens ranges that give appropriate image sizes for the HAARP optics. **Table 1** below compares these possibilities; here Efficiency is a measure of the total light collected by the lens into the image format of the lens (or throughput) , so is proportional to Image Area divided by F number (or $d^2/F\#$). Consideration of the various options leads to the selection of the Pentax 6x7 series as being overall most suitable for this application.

1.4 Telecentric Optics:

The exit ray cones from a commercial fisheye lens are made telecentric by the addition of lenses near the image plane. **Figure 1.2** shows the right-hand part of the ray diagram of **Figure 1.1**, with the addition of two plano-convex elements [B in **Figure 1.2**] to produce a telecentric configuration.
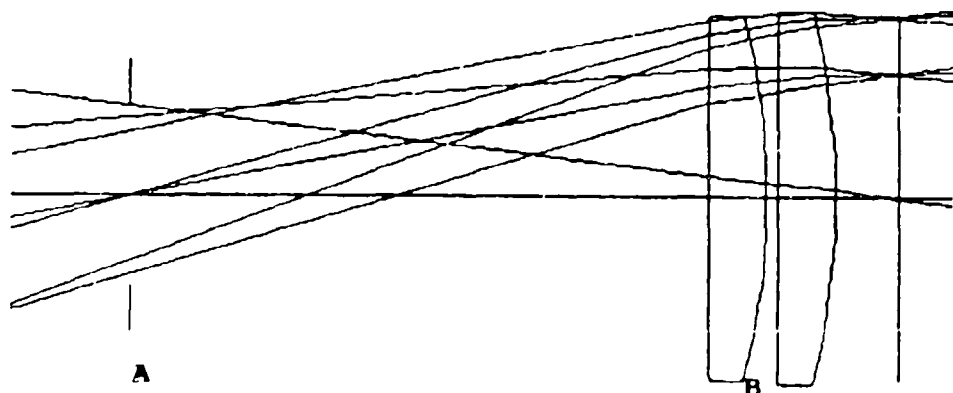
Figure 1.2: Telecentric element configuration

Notes: 1. A single bi-convex lens would result in considerable spherical aberration.
2. A single plano-convex lens would have a very small radius of curvature to achieve the required (small) focal length, and would have considerable spherical aberration (but less than Case 1).
3. Two plano-convex lenses allows reasonable surface curvature, and further reduce spherical aberration compared to Case 2.
4. Three plano-convex lenses would lead to a further small improvement in spherical aberration, but is not justified in terms of reduced transmission and expense.
5. The orientation of the two plano-convex lenses is chosen to share refraction approximately equally between the four surfaces, leading to less aberrations than any other two-element configuration.

Medium Format Lens Comparisons for HAARP Imager

Lens Type	180 deg.	~ 75 deg.	~ 50 deg.	~ 28 deg.	~ 18 deg.	~ 10 deg.	~ 7 deg.	~ 5 deg.
Mamiya 645 (70mm diag.)	24mm/F4.0	45mm/F2.8 (used at F4.0)	80mm/F1.9 (used at F4.0)	150mm/F3.5 (used at F4.0)	210mm/F4.0	500mmF5.6	None	None
Efficiency:	1.00	1.00	1.00	1.00	1.00	0.50	None	None
Mamiya RB67 (92mm diag.)	37mm/F4.5	65mm/F4.5	90mm/F3.8 (used at F4.0)	180mm/F4.5	250mm/F4.5	500mm/F8.0	None	None
Efficiency:	1.37	1.37	1.73	1.37	1.37	0.85	None	None
Pentax 6X7 (92mm diag.)	35mm/F4.5	55mm/F3.5 (used at F4.0)	90mm/F2.4 (used at F4.0)	150mm/F4.0	300mm/F4.	500mm/F4.0	800mm/F4.0	1000mm/F8
Efficiency:	1.37	1.73	1.73	1.73	1.73	1.73	1.73	0.43
Hasselblad (85mm diag.)	30mm/F3.5 (used at F4.0)	50mm/F4.0	80mm/F2.8 (used at F4.0)	150mm/F4.0	250mm/F5.6	500mm/F8.0	None	None
Efficiency:	1.47	1.47	1.47	1.47	0.74	0.37	None	None

Table 1

The Table above compares available commercial "medium format" lenses that give a ~ 3" - 4" image diagonal, consistent with using the largest practical interference filter for maximum light collection. Only lens brands that include a fisheye lens in the line are considered.

If we set the minimum F number for light through the filter at F4.0 (7° beam half angle), then the efficiencies shown are relative efficiencies referenced to the efficiency of a F4.0 lens with a 70 mm image circle (2½" format). Note that some focal lengths are available at lower F numbers than F4.0, but using the lenses at these high apertures increases the cone angle of light through the filter, which necessitates wider bandwidth filters, with consequent transmission of more sky background. As the required filter bandwidth increases as the square of the cone angle, S/N is not improved by using lower F numbers

It may be seen that the Pentax range offers the best overall efficiencies. The Pentax range is particularly advantageous at higher focal lengths, which could be useful in the HAARP program for detailed imaging of heater spot structure.

5

The effective focal length of the two plano-convex elements is chosen to equal the distance from the exit pupil [A in **Figures 1.1, 1.2**] of the lens to the principal plane of the two-lens combination [B in **Figure 1.2**]. Thus the principal rays of all image-forming cones are refracted parallel to the principal axis of the lens.

This is the so-called telecentric configuration. This allows the use of narrow-band interference filters, with a maximum ray angle through the filter determined by the F number of the lens (e.g., 7.1° for F4.0). Note that the image size is slightly smaller than that formed by the primary lens itself.

1.5 Variable Field of View:

Different primary lenses are used to achieve a range of fields of view, but in each case appropriate telecentic elements and spacings must be selected (see **Figure 1.3**). For the HAARP instrument, all primary lenses are in the same commercial series (Pentax 6x7 format), and these are mounted onto telecentric lens housings with the appropriate plano-convex elements and spacings. These combined primary lens/telecentric lens assemblies are then interchangeable, in that they all give the same image size and same telecentric cone angle, so that the following optics is the same for all.

1.6 Accuracy of Telecentricity:

The telecentricity achieved is not perfect, but varies across the image. **Figure 1.4** shows the deviation from perfect telecentricity for the fisheye lens of Figure 1.1, with various combinations of two plano-convex telecentric elements. The zero crossover point can be set anywhere, and is normally chosen to be at about 0.75x the image radius so as to minimize the $\pm$ deviations. (To achieve better telecentricity than the $\pm 1^\circ$ shown would require specially designed aspheric elements.)

6
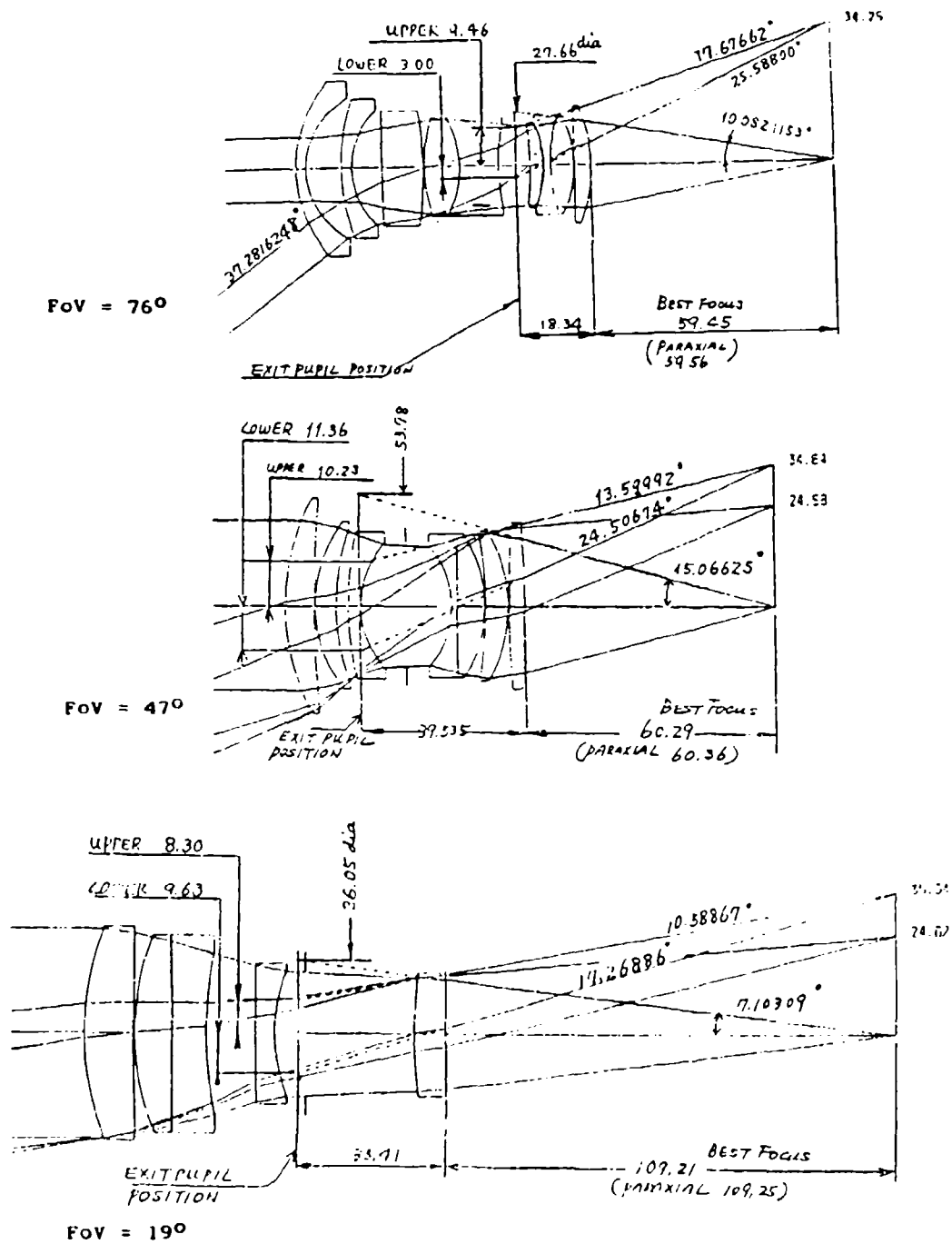
Figure 1.3: Shows typical ray diagrams for some other lenses with smaller fields of view. Telecentric elements may be chosen for each of these lenses (but focal lengths and spacings differ for each primary lens).

Deviation of Exit Cone from Perfect Telecentricity
(Fisheye Lens)
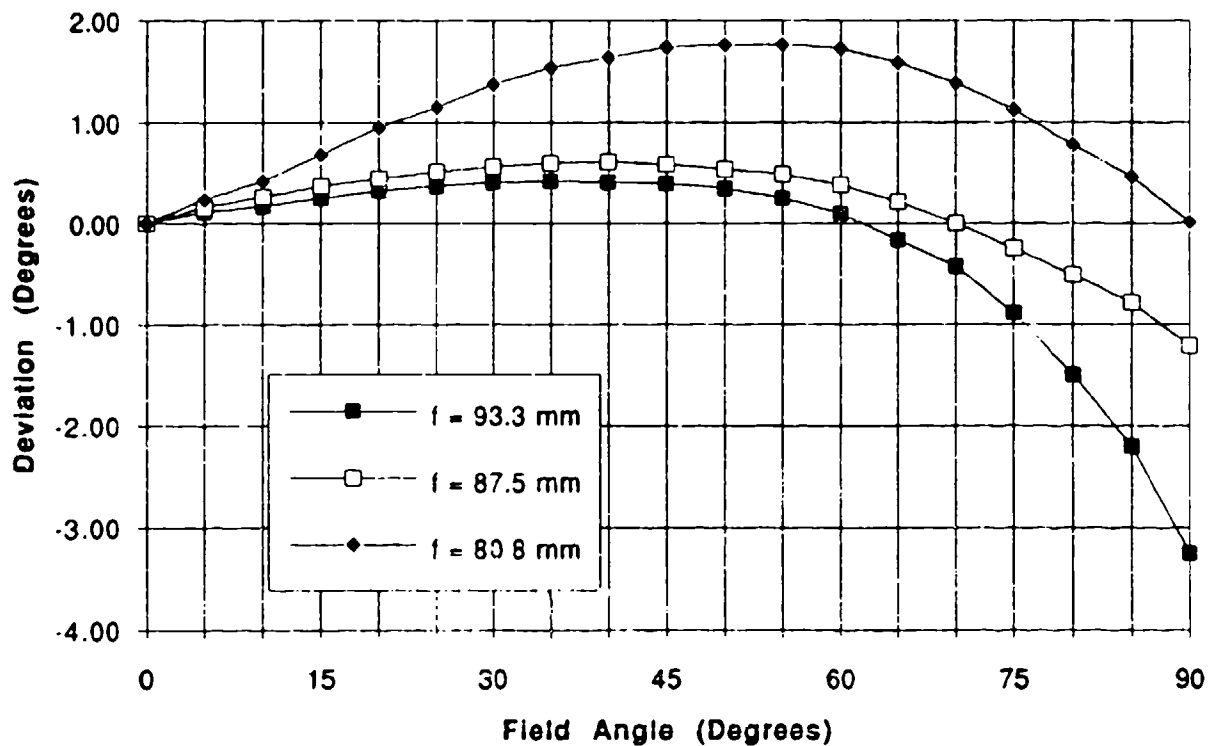
Figure 1.4: Shows deviation from perfect telecentricity for various focal lengths of the combined plano-convex telecentric elements. Appropriate choice of focal lengths (~ 87.5 mm) and spacings limits deviation to $\leq \pm 1$ degree.

1.7 Re-imaging Optics:

The re-imaging optics to the image intensifier is shown schematically in **Figure 1.5**.
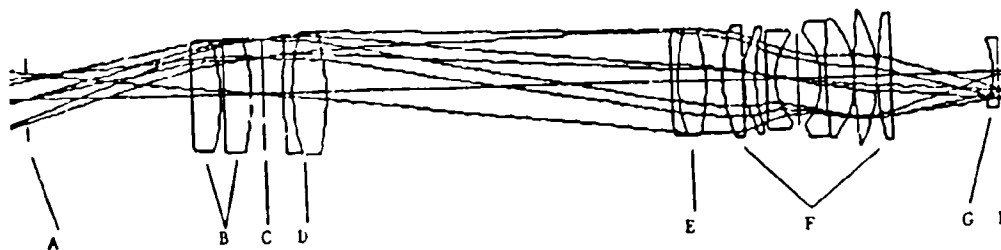
Figure 1.5: Re-imaging optics schematic

The rays making up individual image-forming cones would continue to diverge after the primary focal plane (C in **Figure 1.5**), and portions of each ray cone would be lost from the following optics. Consequently a field lens [D in **Figure 1.5**] is inserted just after the filter position. To minimize aberrations, an achromatic doublet is used in the orientation shown. The focal length of this lens depends on the final image size desired. The camera lens [F in **Figure 1.5**] is placed near the common pupil of all the ray cones from the image. To minimize aberrations (especially field curvature), it is desirable to use the camera lens at near its infinity focus. Consequently a close-up lens [E in **Figure 1.5**] is placed in front of the camera lens, with its focal length chosen to be the same or slightly longer than that of the field lens. Again, to minimize aberrations, an achromatic doublet is used with the orientation shown. The field lens and close-up lens configuration is shown in **Figure 1.6**
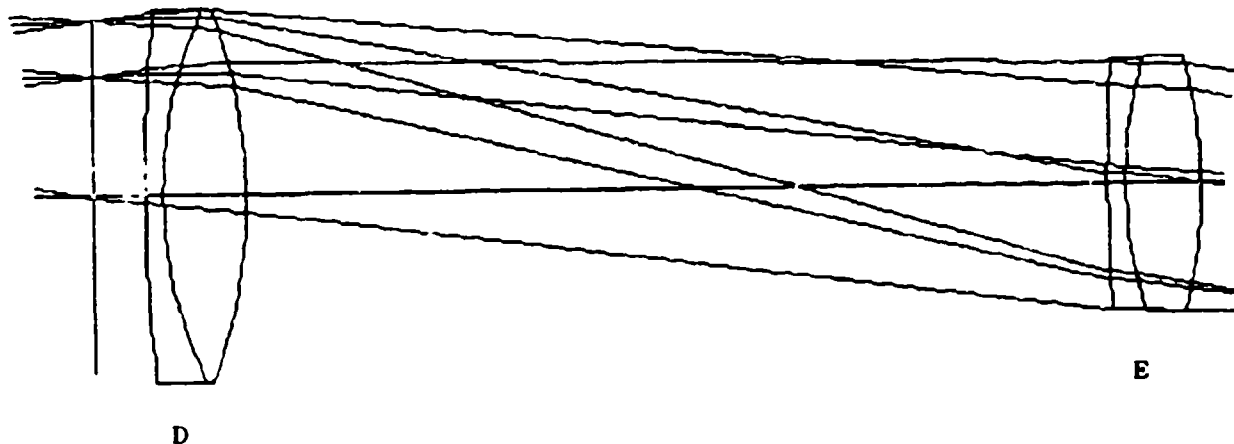
Figure 1.6: Field lens and close-up lens

The focal length (f) of the camera lens [F in **Figure 1.5**] and its separation (L) from the primary image plane is determined by the required final image size (d) on the detector. If D is the primary image diameter, then:

$$d/D \quad = \quad f/L$$

D is fixed by the choice of primary lens format (nominally 92 mm for the Pentax 6x7 medium-format lenses (but this diameter is reduced slightly because of the telecentric elements, see

9

Figure 1.2). Specification of **d** defines f/L. f can then be chosen according to various other requirements, such as:

(i)   Values of **f** available in commercial camera lenses.

(ii)   To minimize optics size, choose a small **f**.

(iii)   To minimize field curvature, choose a larger **f**.

(iv)   To maximize sensitivity, choose **f** so lowest F number lens can be used.

(v)   **f** must also be chosen so that the whole final image diameter can be covered at the chosen F number.

(vi)   **f** should not be so small that L is too small for the procurement of a practical close-up achromat with the required diameter.

Note: Typical example:

1.   **d** $=$ 24 mm (to fit an 25 mm image intensifier).

Then using a Pentax primary lens,

d/D $=$ 24/89 $\sim$ 0.27   (telecentric plano-convex lenses reduce the normal 92 mm image diameter to 89 mm)

If   **f** $=$ 50 mm, then $L \sim$ 185 mm, and

if   **f** $=$ 85 mm, then $L \sim$ 315 mm

If F is the F number of the primary lens, and all of the light is re-imaged to a final image size **d**, then the required F number of the camera lens (so as to collect all available light) is

$$F_{Cam} \leq F \times d/D$$

In the above case with F $=$ 4.5 and d/D $=$ 0.27, a F1.2 camera lens would be required. In general if space allows, it is best to choose the longer focal length camera lens so as to minimize image curvature (see following).

1.8 Field Curvature:

There will always be some residual field curvature. This is reduced if longer focal length camera lenses are chosen, but this is not always possible or desirable. The curvature can be significantly reduced if a field curvature correction lens (a plano-concave element, G in **Figure 1.5**) is placed just in front (1-2 mm) of the final image plane. For maximum resolution, this field curvature corrector is desirable if allowed by physical restraints at the detector. The focal length of the field curvature correcting lens for minimum field curvature was determined by trial and error, and for the HAARP optics is f = -100 mm. A comparison of ray diagrams with and without the field curvature correction lens is shown in **Figure 1.7**
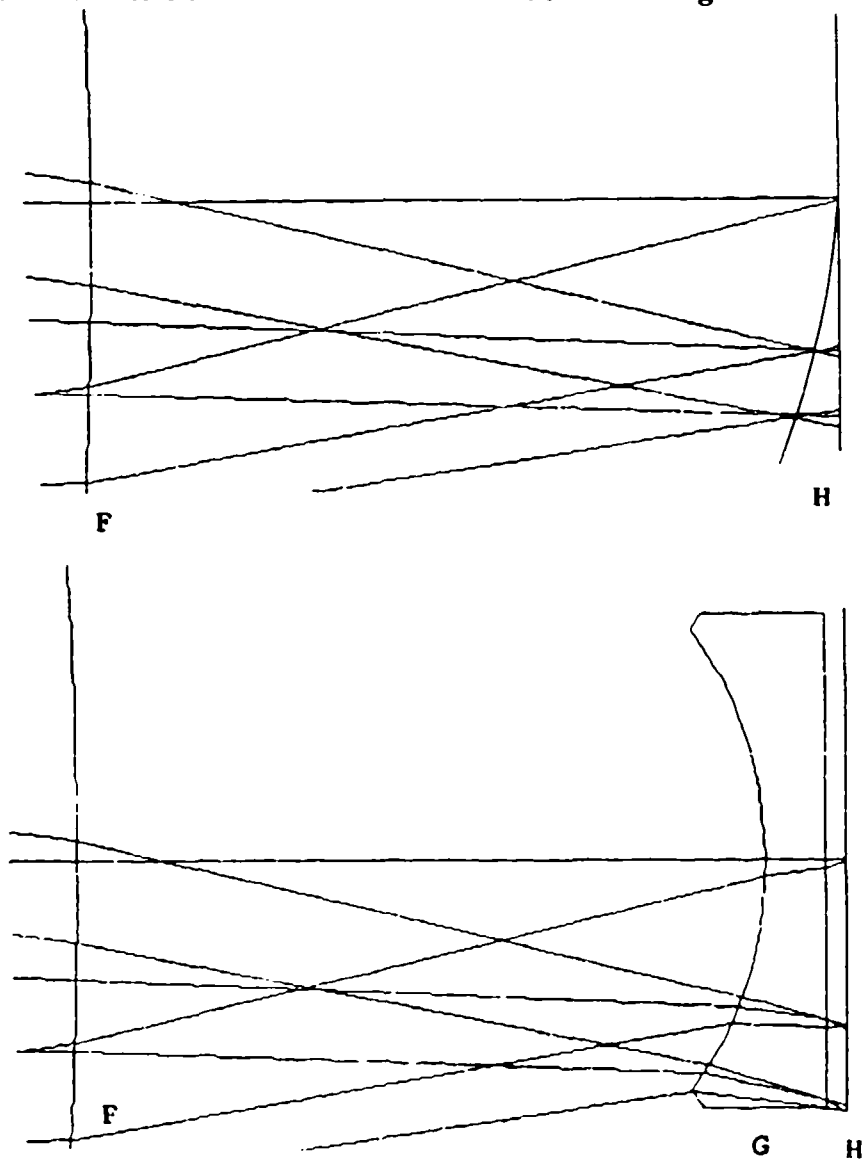
Figure 1.7: Field curvature correction

1.9 Image Intensifier:

The image intensifier used is a 25 mm Gen II Inverter tube **(Figure 1.8)** This type of intensifier is superior to proximity-focused tubes for gain and resolution, and is also considerably less expensive. If near infra-red images were desirable, then a hybrid tube consisting of an Gen III (GaAs) proximity tube coupled to a single-stage inverter tube is also available and interchangeable with the present tube.

The cathode is a thinned tri-alkali (for improved blue quantum efficiency) and the output phosphor is a P20 (for good time response, with a decay time to the 10% level of ~ 1-2 msec).

The intensifier is housed in a custom-designed thermoelectric cooler, that cools just the cathode (to reduce dark noise) but leaves the phosphor at ambient (so as not to increase phosphor decay time). Cooling is to about 20°C below ambient, which gives a 10:1 reduction in thermal dark current.
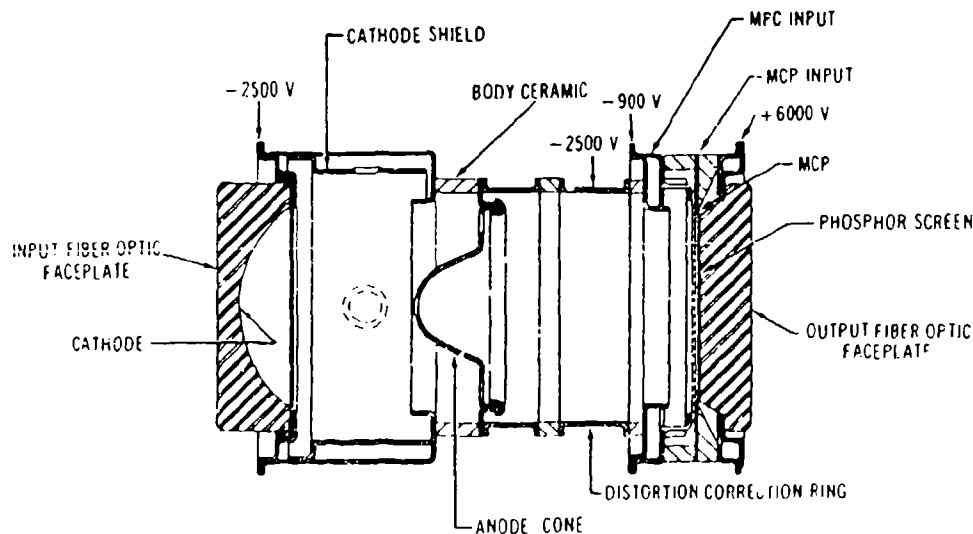
The electrostatic image-inverting generat.. 1 2 image tube

Figure 1.8: Schematic of a Gen Ii Inverter tube

12

1.10 Relay Optics:

The output image from the image intensifier has to be coupled to the CCD, with a reduction in size from 24 mm to 10 mm (CCD size is 10.2 x 10.2 mm). This coupling could have been achieved with either a fiber optics taper or with relay lenses, and there are advantages and disadvantages to each approach.

The relative efficiencies of fiber optics and relay lens depends on the image magnification ($m, \leq 1$) required:

(a) For a fiber optics taper with magnification m, the coupling efficiency is given by
$$T \times m^2 \quad \text{where } T \text{ is the fiber optics transmission}$$

(b) For a non-vignetting relay lens, the coupling efficiency is given by:
$$T / [\, m^2 + 4xF^2 \times (1+m)^2 \,] \quad \text{where } T \text{ is the lens transmission}$$

and F is the F number of the relay lens (system) onto the detector. Usually $m^2 \ll [4xF^2 \times (1+m)^2]$ and may be neglected, giving $T / [\, 4xF^2 \times (1+m)^2 \,]$

In both cases, $T \sim 0.8$, and the following Table compares coupling efficiencies:

Coupling Efficiency of Fiber Optics Tapers and Relay Lenses

m	Fiber Optics	Relay Lens		
		F = 0.7	F = 1.0	F = 1.4
1	0.800	0.090	0.047	0.025
0.5	0.200	0.172	0.086	0.045
0.33	0.087	0.224	0.111	0.057
0.25	0.050	0.256	0.127	0.065

It may be seen that although fiber optics is much more efficient for 1:1 coupling, its advantage quickly decreases as magnification decreases (as is common when coupling from image intensifiers to CCDs), and for $m \leq 0.33$, relay lenses may be as efficient or even more efficient.

There are many other factors that enter into the choice between fiber optics and relay lenses:

(i) If the detector is to be cooled, it may not be desirable to use fiber optics coupling, as the added thermal load will make CCD cooling difficult. The best solution in this case may be to cool the complete CCD camera/fiber optics/intensifier combination.

(ii) If commercial CCD cameras are to be used, it may not be possible to use fiber optics coupling, as most to not come with the option of a fiber optics faceplate on the CCD. Specialty scientific cameras with fiber-optics faceplate CCDs are much more expensive than commercially available cameras.

(iii) Fiber optics may be necessary if there are physical size and/or weight restrictions, as relay lenses are considerably larger and heavier. Similarly, fiber optics will be advantageous if there are shock or vibration conditions.

(iv) Fiber optics will generally reduce resolution more than relay lens systems.

The following considers aspects of relay lens coupling:

For high efficiency and high resolution image transfer, simple lenses or even achromats are unsuitable, as they will suffer from large field curvature (as well as other aberrations). A compound lens (such as camera or enlarging lens) that is designed for close conjugate applications can be used, but generally the effective F number will be high so the coupling will be inefficient.

Typically two lenses are used:

(i) a collimator lens to collect light from the image intensifier output screen;

(ii) a camera lens to image the light onto the CCD detector.

This arrangement allows both lenses to be used at their infinity conjugate, and so maximizes coupling efficiency and minimizws aberrations.

14

Normal camera lenses (e.g. Nikon, Canon) cannot be used in tandem configuration as serious vignetting results. A specially designed collimator lens (that "overfills" the camera lens) is required, such as those available from Rodenstock. Normal camera lenses can then be used as the final imaging lens, though Rodenstock also supplies high-speed camera lenses especially designed for use in tandem configuration with their collimator lenses. (See **Figure 1.9**).

The ratio of focal lengths of the camera and collimator lenses determines the image magnification, as follows:

$$m \quad = \quad \frac{\underline{\text{Focal length of camera lens}}}{\text{Focal length of collimator lens}}$$
$$\sim \quad 0.4 \text{ for the HAARP configuration.}$$

The fastest available relay lens pair is a Rodenstock 100mm/F1.5 coupled to a Rodenstock 42 mm/F0.75.

If F_{cam} is the F number of the camera lens, and F_{coll} is the F number of the collimator lens, then for the relay lens system to have no vignetting, the requirement is that

$$F_{cam}/F_{coll} \quad \leq \quad m$$

For the HAARP configuration, $m = 0.42$ and $F_{coll} = F1.5$, so we require $F_{cam} \leq$ 0.63 The camera lens used (42 mm/F0.75) almost meets this requirement, accepting a F1.8 cone from the collimator lens (thus using 70% of the light collected by the collimator lens). Fast camera lenses like this have very short back focal distances ($\sim$ few mm), so it was necessary to specially design the lens mount configuration for the Photometrics CCD camera.

1.11 Anti-Fogging:

There is always the possibility with cooled detectors that some surfaces may fog or ice up in humid environments. Consequently the capability of dry N_2 flushing has been built into the HAARP instrument. Both sides of the cover glasses in front of the image intensifier, and the front of the CCD cover glass (the back side is in an evacuated environment) can be flushed. Experiments have shown that flushing for $\sim$ 1 minute will rapidly remove any condensation.
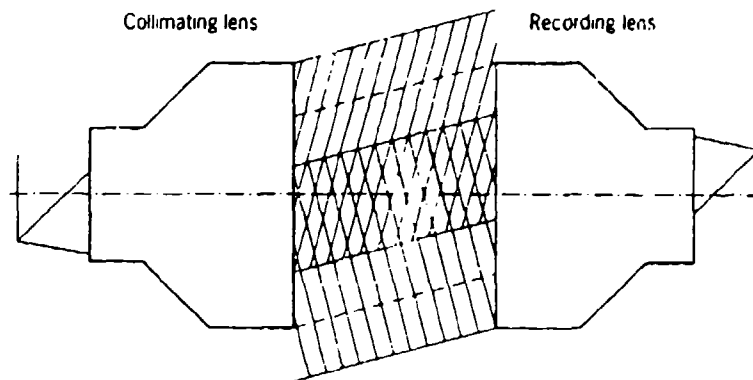
Vignetting in a tandem system
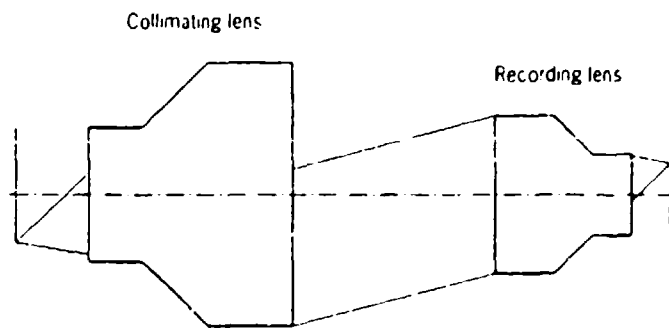
Non-vignetting tandem system with over-
sized collimating lens – see also page 3.40
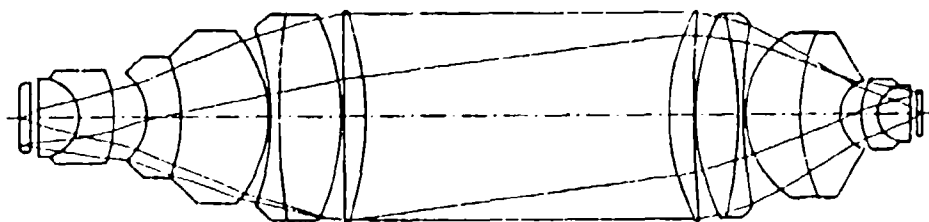
Section of tandem system

Figure 1.9: Relay lens considerations

1.12 Shutters:

There are two shutters in the instrument, one in front of the intensifier lens, and the other between the two relay lenses. This allows independent determination of intensifier and CCD noise characteristics. In addition, there is a light sensor built into the input side of the relay lens shutter. This allows pre-monitoring of the output of the image intensifier; this information can be used to adjust exposure time or high voltage setting of the intensifier, so as to keep the image well placed within the dynamic range of the CCD detector, and to determine if the image intensifier is near its AGC (automatic gain control) mode.

1.13 High Light Protection Light Sensor:

A sensor is built into the front of the filter wheel to measure the ambient light level. If this higher than a preset limit, the image intensifier will not be allowed to turn on.

Chapter 2. Instrument Performance and Calibration

2.1 Advanced Technologies CCD Head

2.1.1 Advanced Technologies Delivered Specifications

Figure 2.2 contains the original performance specifications as set by Advanced Technologies 6/24/92. There were six images supplied with the camera that are useful for reference (**Figure 2.3**):

Low-level resolution image	Hi-level resolution image	Dark Noise image
Lo-Gain Bias	Mid-Gain Bias	Hi-Gain Bias

It was noticed that the Bias settings were unnecessarily high and were limiting the dynamic range of the instrument ($100/4096 = 2.5\%$). In the dark-noise integration, one can notice the intrinsic pattern of dark-noise buildup for these CCDs.

Upon taking images with the HAARP Imager and comparing them to the MIP and ASIPII imagers, it was noticed that the HAARP CCD images were flipped and rotated 90 degrees CW as **Figure 2.1** shows:
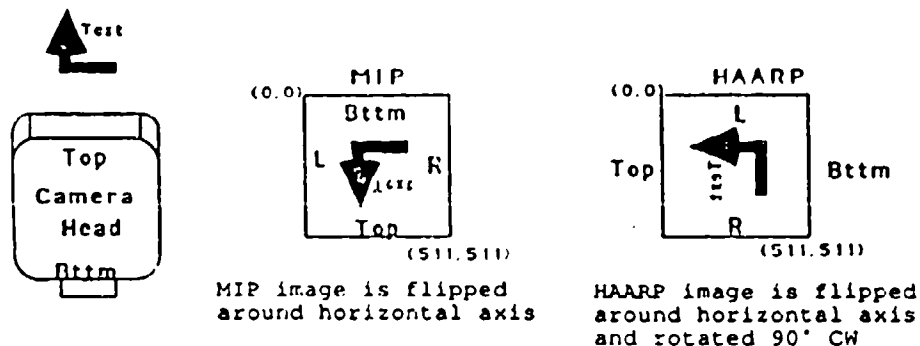
Figure 2.1: CCD Orientations 9/23/92

2.1.2 Advanced Technologies RMA 11/17/92:

The HAARP CCD Head/Electronics was returned to Advanced Technologies to fix the above problems. The CCD was rotated 90 degrees inside the vacuum chamber and the biases were lowered to increase the dynamic range of the ADC. **Figure 2.4** contains the calibration images from this RMA.

CAMERA PERFORMANCE TESTS

Open-Loop Temperature: _____°C   Regulated Temperature: _____ °C

Dark Current ✓MPP ___NON-MPP   Mode : _____ e-/_____ at _____°C at _____ gain factor

Bias Mean Level: _120_ ADU at x4 gain factor

Well Capacity at A/D Converter Limit: _200_ ke- at x2 gain factor

Reduced Full Well Capacity in MPP Mode: ≈200 ke- at x2 gain factor

---

GAIN/NOISE MEASUREMENTS
_____ HZ System
A/D Conversion Factor

Measured at 1X Gain: _85_ e-/ADU   Noise: _52_ e-

✓Verified at 2X Gain 49.5   Noise: _40_ e-

✓Verified at 4X Gain 24.4   Noise: _26_ e-

FINAL VACUUM TESTS

___✓ Final Pump and Bake Cycle Performed (_16_ Hours)
_____ Final Water Vapor Test Performed
_____ Final Helium Leak Test Performed
_____ Vacuum Valve Packed With Crease
_____ Final Vacuum Leak Check Performed

FINAL VOLTAGE SETTINGS

Vod___   +Vsh___   -Vsh___
Vrd___   +Vvp___   -Vvp___
Vog___   +OR___   -OR___

Comments _____

_____

_____

_____

_____

_____

_____

Testing performed by _____   Date _6-24-92_

Approved by _____   Date _____

Figure 2.2a -- Advanced Technologies CCD Specifications 6/24/92
Performance Tests

19

ATC-5 TEST REPORT

CUSTOMER _____ KEO _____
ATD JOB NUMBER _____ AT220 _____
CAMERA HEAD SERIAL NUMBER _____ 2 _____
ELECTRONICS UNIT SERIAL NUMBER _____
COOLING UNIT SERIAL NUMBER _____ NA _____

FINAL ASSEMBLY AND TEST BY _____ (signature) _____
DATE _____ 6-24-92 _____

CCD TYPE _____ PMS16A _____
CCD SERIAL NUMBER _____

CCD PARAMETERS

VOD1	21.7
VOD2	26.0
VRD	12.5
VOG	0
VSL	-4.88
VSH	+5.13
VPL	-2.53
VPH	+3.05
VSWL	-4.97
VSWH	+5.36
VTGL	-7.55
VTGH	+3.71
VRH	7.60
VRL	-.02
VIG	NA
VID	NA
VSUB	-9-

R_{LOAD} 3.01 K

Figure 2.2b -- Advanced Technologies CCD Specifications 6/24/92

Voltage Settings
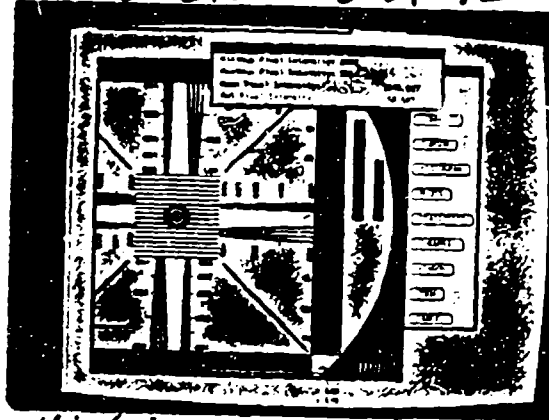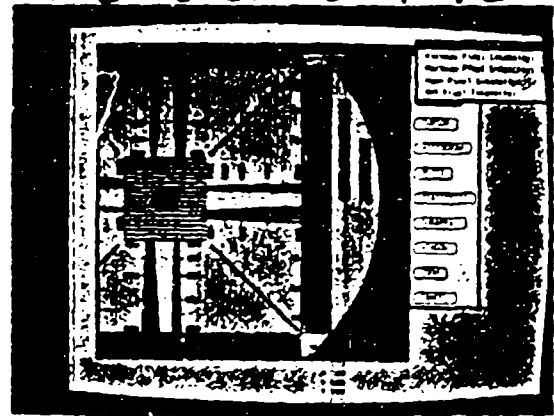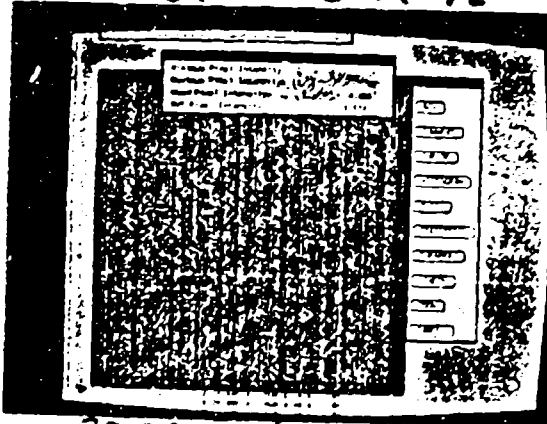
20
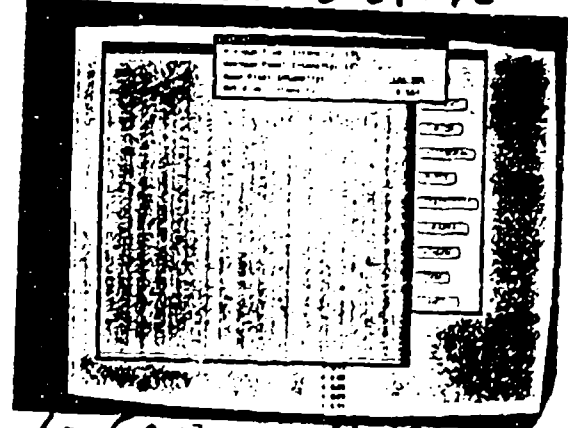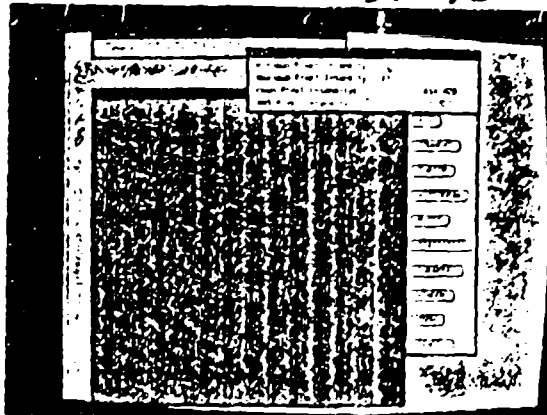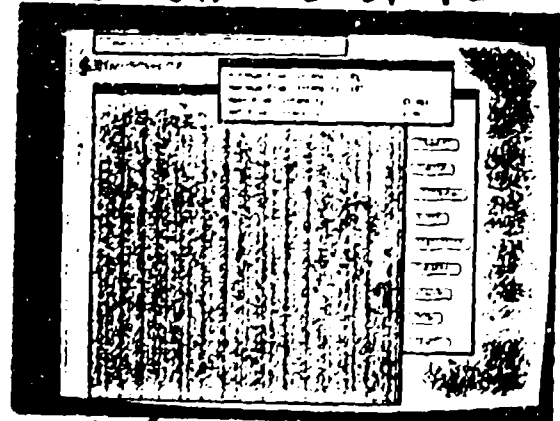
Figure 2.3 -- Advanced Technologies Performance Images 6/24/92
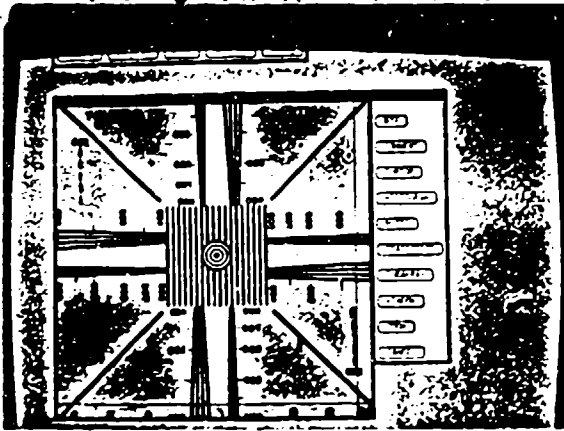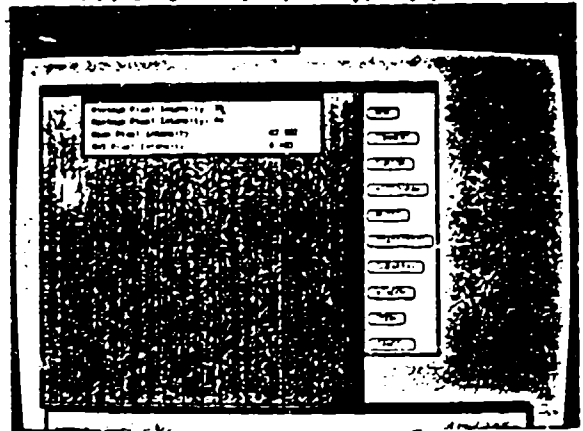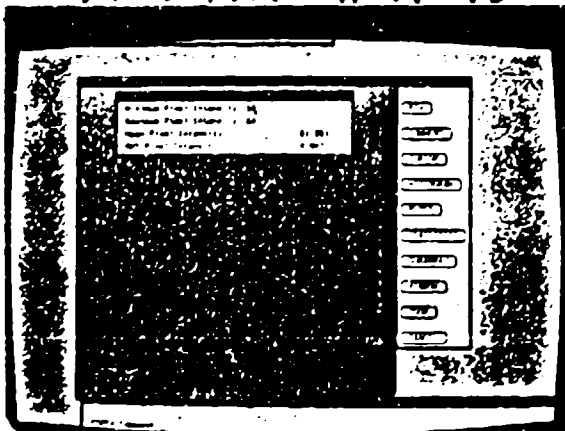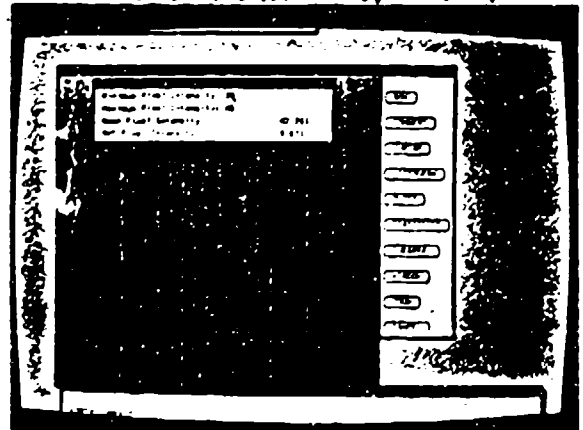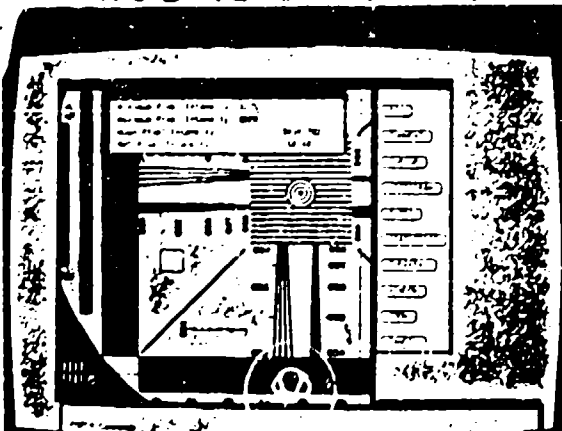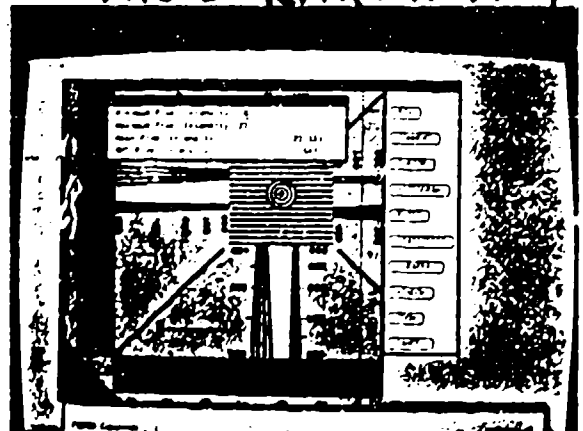
Figure 2-4 -- Advanaced Technologies RMA Images 11/17/92

22

2.1.3 CCD Flatfield

A flat-field image was acquired on the PM516A to check for non-uniformity in the CCD's performance. The following setup was used:

CCD Head

13 Feet

PM516

w/no Lens

1. Light Source
2. Aperture
3. Shutter

1 2 3
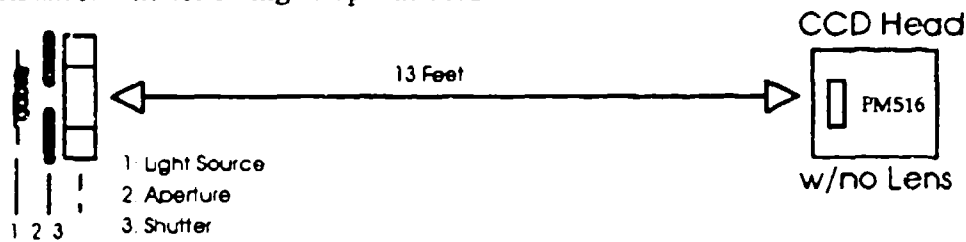
Figure 2.5 shows a 1-second exposure taken with this setup. The first image is the flat-field looking at the full dynamic range of the CCD. The second image expands the detail to look at the CCD variations in more detail.
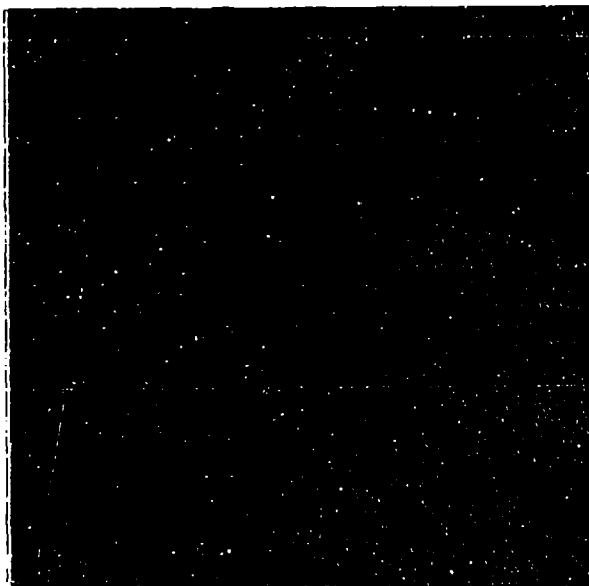
[Full Dynamic Range: Black = 0, White = 4095]   [Stretch: Black = 1600, White = 1650]
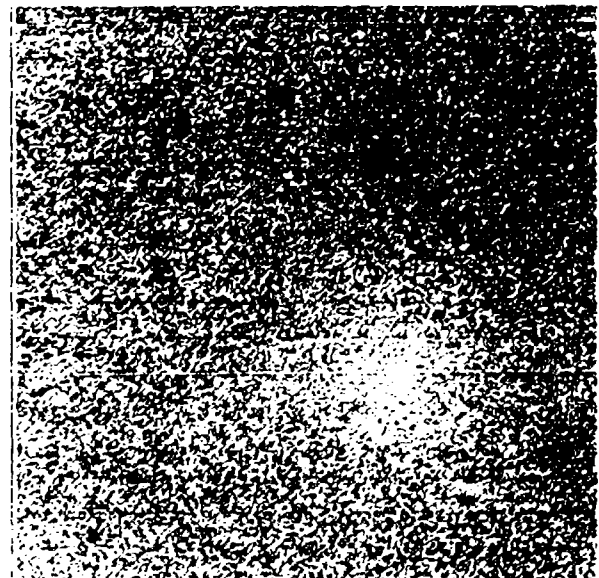
Figure 2.5 -- CCD Flat-Field Images

There are dust shadows on this image that result from very small particles of dust on the vacuum window. These shadows do not appear on the resultant image with a lens as the window is not in an image plane. **Figure 2.6** shows a plot through the image of column 261:
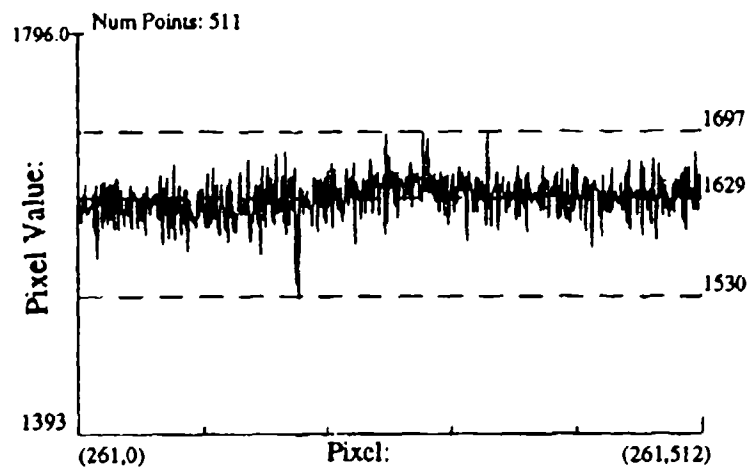
Figure 2.6 -- Column Plot from Flat-Field Image

From the plot in **Figure 2.6** and the expanded flat-field image, we can see that there are certain patterns that are intrinsic to the CCD that appear when looking at the flat field in great detail. Notice the periodic horizontal line pattern that can be seen.

2.1.4 CCD Resolution:

A resolution chart was projected onto the CCD using the 42mm Rodenstock relay lens (and the back shutter of the instrument for exposure control. The iris of the back shutter was closed to about 1/4 aperture. Since the Rodenstock has a maximum image size of 11mm which is smaller than the 14.4mm diagonal of the CCD, and in this case the lens is completely filled, you can notice the blurring of the lens at it's edges in the resolution image in **Figure 2.7**. This does not affect our images as the intensifier image does not completely fill the 42mm lens image circle.

The theoretical limit of the CCD resolution is determined by the number of pixels in the image. The PM516A uses 512 pixels across it's image area, each with a dimension of $20 \times 20 \mu m$ giving a 10.2mm CCD size. Thus, the maximum resolution for a line-pair (black/white) for this CCD is 256 line-pairs or 25 lp/mm.
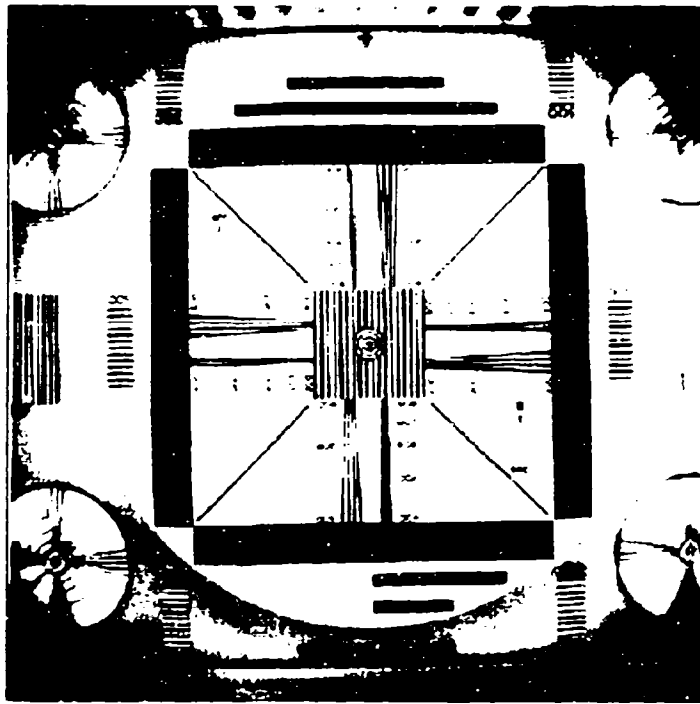
Figure 2.7 -- CCD Resolution Image

2.1.5 CCD Bias Settings

The Bias voltage is a 'zero-offset' voltage that is applied to the summation-amplifier on the CCD before it is converted at the Analog to Digital Converter (ADC) and assures that we are operating above the threshold of the electronics. The bias is somewhat arbitrarily set by Advanced Technologies along with the other clocking voltages (documented in Sections 2.1.1 and 2.1.2) and should not be changed in the field.

Initial Bias Settings (6/24/92) were: LO: 105   MID: 103   HIGH: 114

These were lowered as discussed in Section 2.1.2 to improve the dynamic range of the instrument:

RMA Bias Settings (11/7/92):   LO: 42   MID: 42   HIGH: 61

Biases were closely watched during the calibration of the instrument as there seems to be some fluctuation in their values. Bias images were archived during the Mean-Variance calibration (Section 2.1.7) and were found to be:

MV Bias Readings (7/13/93):   LO: 13   MID: 11   HIGH: 23

25

Figure 2.8 -- Mid Bias image from 7/13/93 (Tccd = -29.5C)

(stretched: black = 10, white = 15)

Figure 2.8 shows an expanded image of the bias at Mid Gain. As in the flat-field image, we again see horizontal patterns that are intrinsic to the CCD readout.

A column plot from a HI CCD gain Bias image in **Figure 2.9** also shows the horizontal patterns intrinsic to the CCD.

COL Plot:
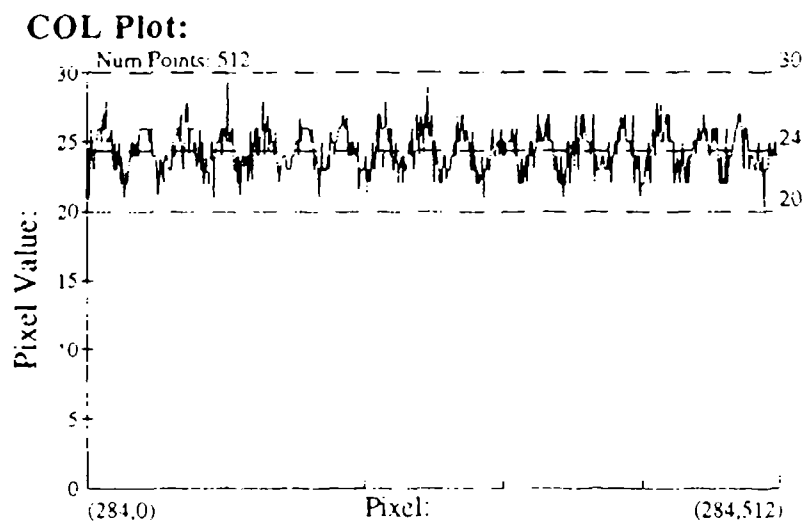
Figure 2.9 -- Column plot from HI Bias image (8/27/93)

26

Bias Readings (8/27/93):   LO: 15.74   MID: 13.2   HI: 24.48

On 9/1/93, we looked at the BIAS levels verses time. The following results were found:

Powered Down/Up the Camera and take BIAS readings repetitively

CCD Temp	LO Bias	MID Bias	HI Bias
-28.5C	1.1 +/- .8	0.19 +/- .5	8.23 +/- 1.6
-29.5C	2.78 +/- .8	1.03 +/- 1.0	9.77 +/- 1.6
-29.5C	3.61 +/- .9	1.73 +/- 1.1	10.8 +/- 1.6
-29.5C	5.45 +/- .8	3.26 +/- 1.2	12.6 +/- 1.6

Powered Down/Up and found MID Bias: 5.42 +/- 1.2

Let Camera sit for 20 minutes and take BIAS readings repetitively

CCD Temp	LO Bias	MID Bias	HI Bias
-26.5C	10.4 +/- .8	8.2 +/- 1.2	18.5 +/- 1.6
-26.5C	10.6 +/- .8	8.4 +/- 1.2	18.7 +/- 1.6
-26.0C	10.9 +/- .8	8.4 +/- 1.2	19.0 +/- 1.6

These results show quite a bit of drifting in the BIAS levels until they seem to settle down to a final value with time. This is probably due to the temperature inside the CCD Electronics head where these voltages are set. The fact that the BIAS level is slightly different for each session points to three things:

    (i) BIAS levels should be watched closely and recorded intermittently during data acquisition sessions.

    (ii) If possible, the instrument should be given time to stabilize (~20 minutes) before data acquisition starts.

    (iii) The overall trend of BIAS levels migrating down from ~40 (11/92) to ~13 (8/93) is cause for some concern and should be watched closely. If the BIAS levels keep decreasing from these levels (approaching 0), the camera should be returned to Advanced Technologies for a new calibration.

2.1.6 CCD Dark Noise

The PM516A CCD is powered in Multi Pin Phased (MPP) mode which decreases the dark count by about a factor of 30. This is achieved by using a boron implant in the CCD wells. **Figure 2.10** shows a dark integration image with a CCD temperature of -25°C which is a normal operating temperature of the HAARP imager. Notice the characterictic pattern of dark noise integration intrinsic to the CCD chip.
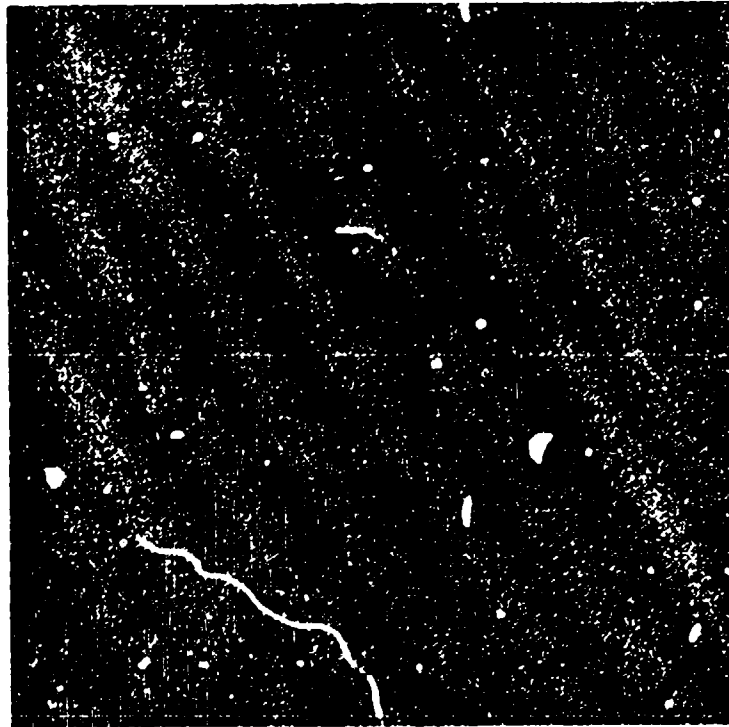
CCD Temp: -25.0C   Exposure: 4 minutes   HI Gain [25e-/ADU]   1x1 Binning
Scale: Black 80 ADU   White 225 ADU

Figure 2.10 -- Dark Noise Integration

Temperature Dependence

Dark Noise for the HAARP camera head was measured verses temperature. Cooling the CCD reduces the dark count by about a factor of 10 for every 20C temperature reduction. The following data for CCD dark noise acquisition was taken with the CCD gain set at HI and using 1x1 binning:

Temp (C)	Exposure (sec)	Mean (ADU)	RMS (ADU)	Mean - Bias	Normalized Mean	Electrons /sec
31	20	1953	269	1930	23160	2412.50
-4	120	473	69	450	900	93.75
-12	120	257	37	234	468	48.75
-19	240	250	36	227	227	23.65
-25	240	147	21	124	124	12.92

This curve is shown in **Figure 2.10**. Examing the above results confirms the rule of thumb for cooling viz: a factor of 2 decrease for every 6°C, or a factor of 10 decrease for every 20°C.
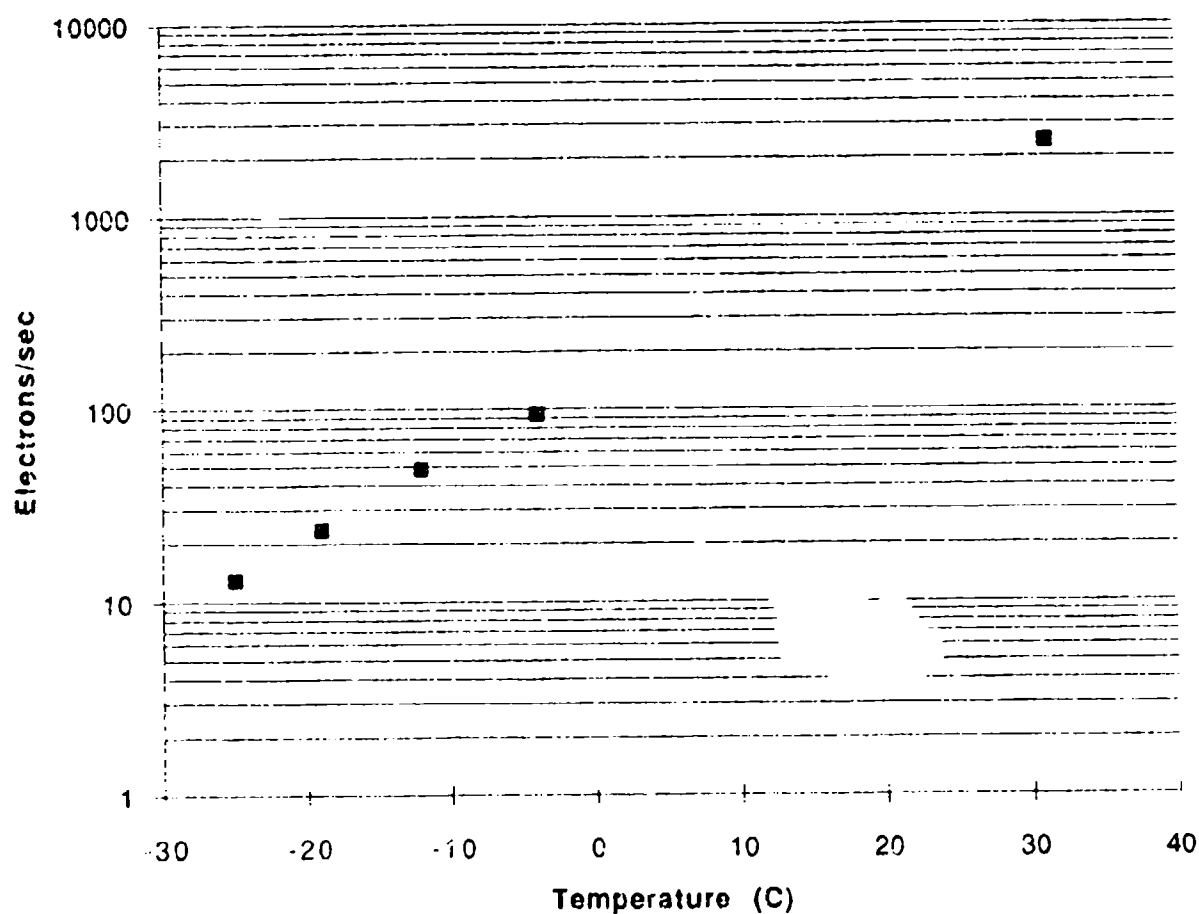
Figure 2.11 -- CCD Dark Count .vs. Temperature

29

2.1.7 HAARP Mean-Variance Calibration

The electron charge in the CCD wells creates a voltage that is amplified by the CCD electronics and converted into digital units using a 12 bit Analog to Digital Converter (ADC) at a pixel rate of 1.1MHz. The Mean-Variance calibration is a method used to determine the actual Analog to Digital Converter gain and noise characteristics. This indicates how many electrons in the CCD well correspond to an Analog to Digital Unit or ADU. The basic principle of this calibration is as follows:

Linear ADC conversion: $ADU = Gain \times N_e$ where N_e = number of electrons

RMS noise of conversion: $\sigma = Gain \times \sqrt{N_e}$

Variance is defined as RMS^2. Therefore, the ratio:

$$\frac{Variance}{Mean} = \frac{(Gain)^2 \times N_e}{Gain \times N_e}$$

The ratio of Variance / Mean gives the gain in units of ADU/electrons. The ratio of Mean / Variance therefore gives us:

electrons/ADU

Advanced Technologies Calibration Method:

Four images are collected with the CCD camera: two independent Bias images and two independent Flat-field images. To do this, the KEO lab was set up as:

30
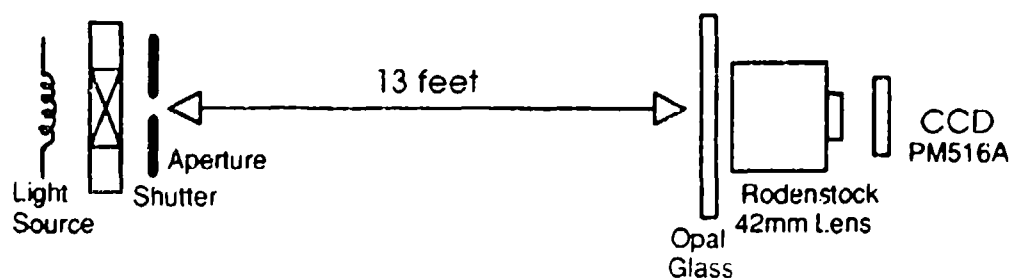
The following definitions were made:

Images taken: **Bias1, Bias2, Img1, Img2**

MEAN:	Mean of (**Img1** - **Bias1**)
2IMG_RMS:	Rms of (**Img2** - **Img1**) [Photon Shot-Noise of images]
VARIANCE:	**2IMG_RMS^2 / 2** [Variance of one image]
GAIN:	**MEAN / VARIANCE** [units: electrons/ADU]
2BIAS_RMS:	Rms of (**Bias2** - **Bias1**)
READ NOISE:	(**2BIAS_RMS** / sqrt(2)) X **GAIN**

Results of 7/12/93 Calibration for HAARP CCD Head:

CCD Head: ATC-2 AT220 SN:2   Temperature: -31.0°C

Low Gain:

Mean: 1432.90   ImgRMS: 5.6   Variance: 15.68   BiasRMS: .969

MidGain:

Mean: 2627.06   ImgRMS: 10.32   Variance: 53.25   BiasRMS: 1.108

HiGain:

Mean: 2140.97   ImgRMS: 13.06   Variance: 85.28   BiasRMS: 1.327

CCD Gain	Gain (e-/ADU)	Read Noise (e-)
Low	91.38	62.61
Mid	49.33	38.65
Hi	25.10	23.55

Bias Statistics T = -26°C

Gain	Calibration Image	Mean	RMS
Low	BiasLow.Img	13.22	0.616
Mid	BiasMid.Img	11.02	0.788
Hi	BiasHigh.Img	22.98	0.913

31

In HI and MID gains, the ADC saturates before the CCD well at 4095. At LO gain, the CCD saturates at 2395 (not including the BIAS level). This gives us the full-well potential of our CCD:

$$2395 \times 91\ e\text{-} \implies \textbf{217,945 e- PM516A full well depth}$$

At MID gain, the ADC saturates at 200,165 e- or very nearly the full well depth of the CCD. At HI gain, the ADC saturates at 101,850 e- or about half of the CCD well depth.

For 1x1 binning, then, we can characterise the use of these gains. When we want to match the dynamic range of the CCD most closely with the ADC, we should operate at MID gain. When we are counting very few photons and want the greatest sensitivity of the instrument, we should operate at HI gain, understanding that the dynamic range of the instrument is halved. However, when struggling for every photon, dynamic range is not usually an issue.

LO gain is setup for use in larger binning situations. We typically collect data in 2x2 binning to conserve on acquisition time and archival space. The summation well on the PM516A has four times the well-depth of a pixel, so that when we bin 4 pixels together (2x2), the summation well can handle the increased dynamic range (870Ke-). LO gain has a dynamic range (in 2x2 binning) of 4095 x 91e- which is: 370Ke-. This is roughly double the dynamic range of the MID gain, but one should note that for 2x2 binning, the ADC will still saturate before the summation well (4 pixels deep), so the full dynamic range of 2x2 binning is not realized.

These gains were confirmed radiometrically using a C14 Light source for comparison. A sequence of images using the 486.7nm filter and the intensifier at a constant gain were taken for each of the CCD gains and added together giving:

	HI Gain	MID Gain	LO Gain
Image stats:	15428 ADU	7515 ADU	4089 ADU
CCD Gain:	25e-/ADU	49e-/ADU	91e-/ADU

	MID/LO	HI/LO	HI/MID
Ratio of stats:	1.84	3.77	2.05
Ratio of Gains:	1.86	3.64	1.96

The correlation between these two measurements is extremely close as is to be expected.

2.1.8 Centering the image in the CCD

Because of mechanical tolerances and the difficulty of physically centering the CCD inside the vacuum chamber exactly along the optical axis of the instrument, it was noticed during calibration that the image was not centered in the CCD. From the images taken during the first part of the calibration, it was decided that the image needed to be moved about 20 pixels to the left, or about .015".

CCD Housing Modification: The CCD housing was modified to allow the CCD head to move inside and two access holes were drilled in the front of the housing to access the mounting screws on the CCD head cover plate.

CCD Head Modification: The cover plate holes were slotted to allow movement of the CCD head with respect to the cover plate thus moving the orientation of the CCD with respect to the optics axis (the Rodenstock 42mm lens screws into the cover plate).

To adjust the position of the image in the CCD, remove the CCD head from the CCD housing. Remove all the 4-40 screws from the CCD cover plate on the front of the CCD head except the two horizontal screws as shown in **Figure 2.12**.

Looking at the front of the CCD Head

—Remove screws

—Leave two screws tightened

CCD
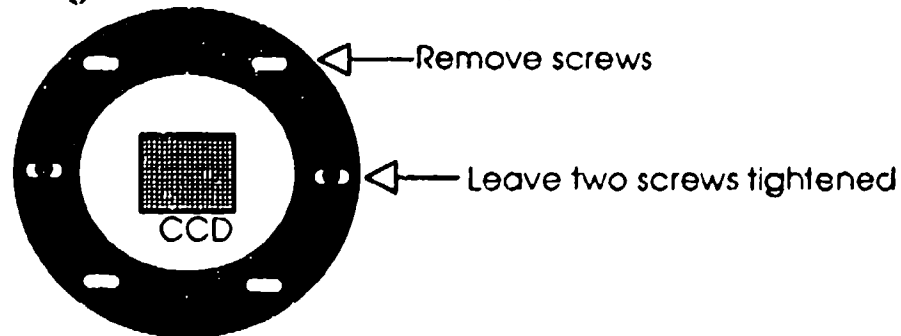
Figure 2.12 -- CCD cover plate screws

With just these two screws holding the cover plate to the CCD head, re-install the CCD head into the CCD housing and take an image. Using the two access holes in the front panel of the CCD housing, slightly loosen the two screws and move the CCD head in the desired direction and re-tighten the screws. Take another image. Repeat this process until the image is centered, and tighten the two screws down tight. Remove the CCD head and tighten in the remaining 4 screws in the cover plate. Re-install the CCD head in the imager.

2.2 Relay Lens Optics

As discussed in Section 1.10, a Rodenstock 100mm collimator lens is placed in front of the intensifier output image and is coupled to a Rodenstock 42mm camera lens which re-images onto the CCD. Rodenstock computed the Modulation Transfer Function for the HAARP setup (**Figure 2.13**) using three different spatial frequencies (95 lp/mm, 48 lp/mm, and 24 lp/mm). The results are shown in **Figure 2.14**.
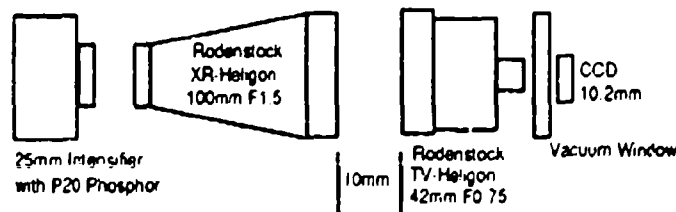
Figure 2.13 -- HAARP relay lens configuration for Rodenstock measurements

There is an intrinsic loss of light from the phosphor in the lens coupling determined by the solid angle subtended by the collimator lens. The plot from Rodenstock gives an angle of 8.53 for the maximum from the focal plane to the edge of the image. Since light from the phosphor exits over a full 180 degrees, the lens at F1.5 only uses 10% of the photons emitted from the intensifier output. From the Rodenstock plot, one can see that there is also vignetting in the lens combination of 7% (98 - 91).

MODULATION-TRANSFER FUNCTION FOR

XR-Heligon 10Cmm F 1.5 ,draw.no.3801.257.20	ON 4307 - 9001
+ TV-Heligon 42nm F 0.75,draw.no.3801.212.22	E - -0.047
Phosphor : P 20 Separation : 10 mm	

BETA'= -0.4242　　　　BLENDENDURCHMESSER = 55.02　　●
SCALE　　　　　　　　　F-STOP DIAMETER

ORTSFREQUENZ: 95. 48. 24. 1/mm
SPATIAL FREQUENCY: ⟋　　　　　　　　　　F-NUMBER

　　　(λ)
p20p(T) LAM 446.0nm 498.0nm 550.0nm 602.0nm 654.0nm
(Relative) BEW (%) 3.6 16.8 65.4 95.3 87.6 43.8 25.3 13.6 5.8
(Output)

x = BEUGUNGSTHEORETISCHER WERT　　————————— SAGITTAL
　　DIFFRACTION LIMITED VALUE　　- - - - - - - - - - MERIDIONAL

Y (Dist. from center of Int) -3.75　　-6.26　　-8.76　　-10.64　　-12.51
Y

Y' = 0 (MITTE)　　　　1.591　　2.651　　3.710　　4.497　　5.258
Y' = 0 (AXIS)
　(Dist. from center of CCD)

WINKEL (GRAD)　　　　2.58　　4.29　　5.99　　7.26　　8.53
ANGLE (DEGREE)

HELLIGKEIT (%)　　　　98　　　97　　　96　　　96　　　91
LIGHT INTENSITY (%) (Relative Vignetting)

VERZEICHNUNG (O/OO) 0.23　　0.27　　-0.28　　-1.84　　-8.16
DISTORTION (0.1%)
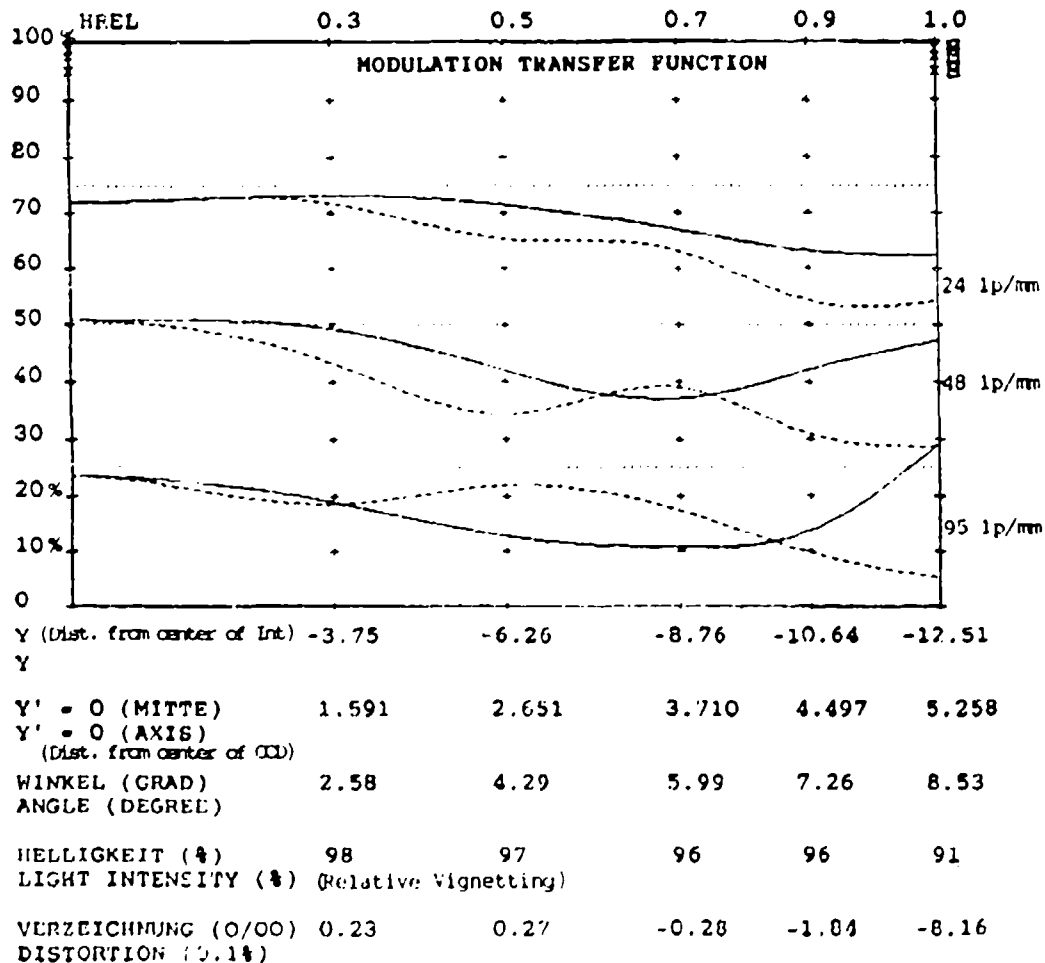
Figure 2.14 -- Rodenstock Relay Lens MTF Calculations

2.3 Shutter Modifications

The HAARP imager uses two Melles-Griot shutters. During calibration of the imager, the back shutter started failing intermittently. KEO examined the rear shutter and found what we determined to be a flaw in the shutter design. The coupling ring tends to rise in the upper right hand side when the relay arm pulls the ring CCW (opening). This rising action causes friction in the mechanism and creates a starting inertia that the relay can not always overcome. It was found that applying a very light amount of pressure to the outside part of the ring at this point (see Figure 2.15) relieves the friction and allows the shutter to operate freely. It was noticed that there was wearing in the ring and metal flakes where the ring is held down by the washer.
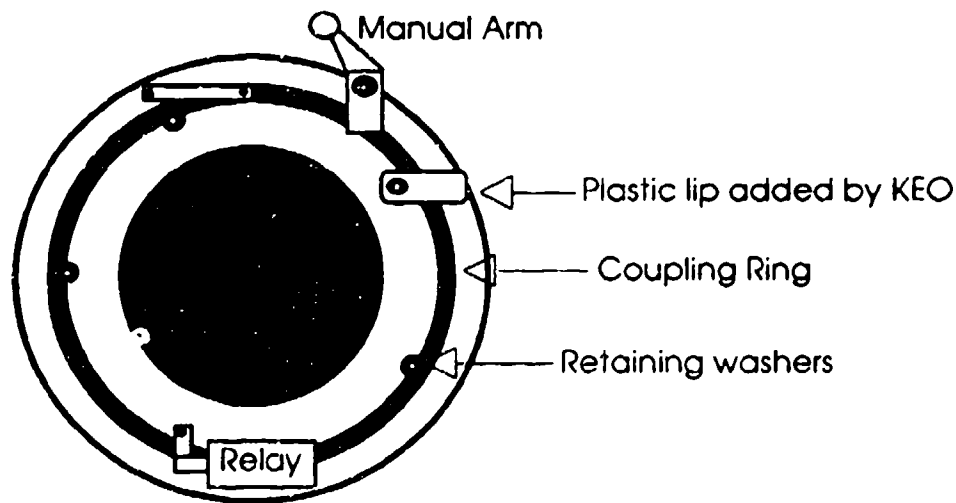
Figure 2.15 -- KEO Modification to Melles-Griot Shutter

To fix this, KEO fabricated two plastic lips (one for each shutter) to replace the metal washers. The plastic lip fastens down in the same position as one washer and extends over the coupling ring applying a little pressure on the ring. This seems to have solved the shutter failures. In the future, teflon holders could be made if the plastic lips become unreliable.

36

2.4 PhotoDiode Calibration

The HAARP imager has a photo-diode placed in the back shutter (SH2) of the instrument to measure the average brightness of the intensifier phosphor. This measurement gives us an idea of how bright the image is and, more importantly, whether the image intensifier is operating in AGC (automatic-gain-control) mode and thus not operating with linear gain. A typical image intensifier output curve is shown in Figure 2.16.

The HAARP imager uses a EG&G VACTEC VTB9413B photodiode. This photodiode has an IR rejection filter on it and has a spectral response (peak is 580nm; range is 320nm - 720nm) similiar to the output of the intensifier P20 phosphour. A photodiode was chosen for this application because of it's relatively fast rise/fall times.

The photodiode output is amplified by an AD515 inside the shutter housing (see Section 3.11), and then to the KEO Interface board where it's gain can be adjusted using the potentiometer VR4. This potentiometer is used to set the output voltage going into the controller's ADC to peak near 5V when the intensifier is in AGC mode. Thus, by reading this ADC value, the controller can monitor whether the intensifier is operating in it's linear range or not. (It turns out that for auroral applications, a typical image does not push the intensifier output anywhere's near the AGC limit).

Image Intensifier Gain Characteristics
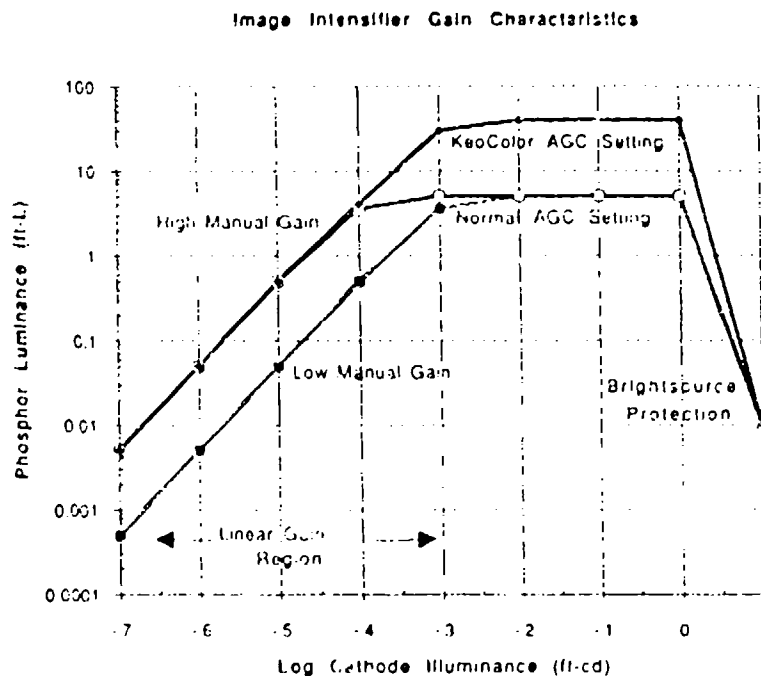
Figure 2.16 -- Typical Image Intensifier Output Curves

The MIP imager used a VTB9413 photodiode, which has a glass window instead of the IR rejection window. This photodiode has a much higher output than the HAARP imager's VTB9413B. To accomodate this difference, a 5.11KΩ resistor had to be added in parallel to the 10KΩ resistor R11 on the KEO Interface board to increase the gain for the AD515 output (see the schematic in the Hardware Chapter). The output at very low light-levels from the AD515 was found to be greater than 0 volts which produces a negative voltage at the input to the ADC (producing 0 ADU's). This limits the photodiode's very low-light level resolution as can be seen in the calibration data. A zero-offset potentiometer could be added to the AD515 amplifier card if necessary.

The following is the Photodiode calibration data for 8/27/93:

Intensifier Gain: Minimum

Setup Gain: Min	Relative Illuminance	AD515 Output (mV)	ADC Input (V)	ADC Ouput (ADU)
Int Off:		2.1	-0.056	0
Shutter Clsd		3.8	-0.084	0
Shutter Open	0	3.3	-0.076	0
LS #10	1	0.7	-0.33	0
LS #9	2	-3.5	0.039	2
LS #8	4	-10.8	0.162	9
LS #7	8	-27.7	0.45	24
LS #6	16	-58	0.966	51
LS #5	32	-122	2.053	108
LS #4	64	-207.3	3.5	185
LS #3	128	-242.6	4.1	217
LS #2	256	-258.9	4.38	231
LS #1	512	-268.6	4.54	240
LS #0	1024	-274.6	4.64	245

Intensifier Gain: Maximum

Setup Gain: Max	Relative Illuminance	AD515 Output (mV)	ADC Input (V)	ADC Ouput (ADU)
Int Off:		2.1	-0.057	0
Shutter Clsd		3.9	-0.085	0
Shutter Open	0	2	-0.54	0
LS #10	1	-36.1	0.592	31
LS #9	2	-101.2	1.699	90
LS #8	4	-196.2	3.31	175
LS #7	8	-250.6	4.24	223
LS #6	16	-268	4.53	239
LS #5	32	-279.3	4.72	249
LS #4	64	-287	4.86	255

The plots in **Figure 2.17** show the curves for the above data. Notice the linear region and the gradual approach to the AGC mode.
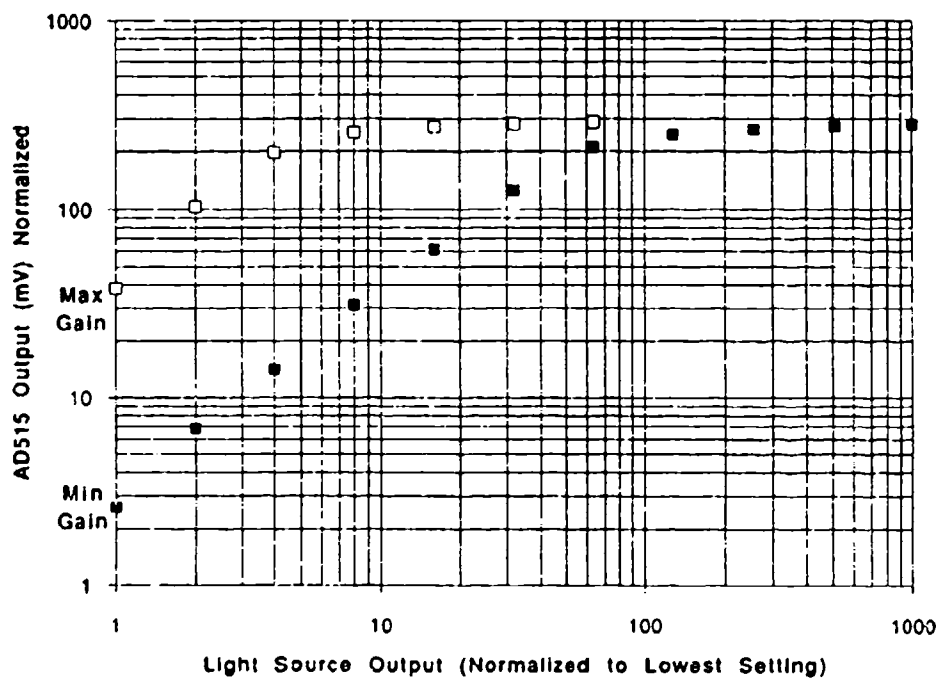
Figure 2.17a -- PhotoDiode Output: AD515
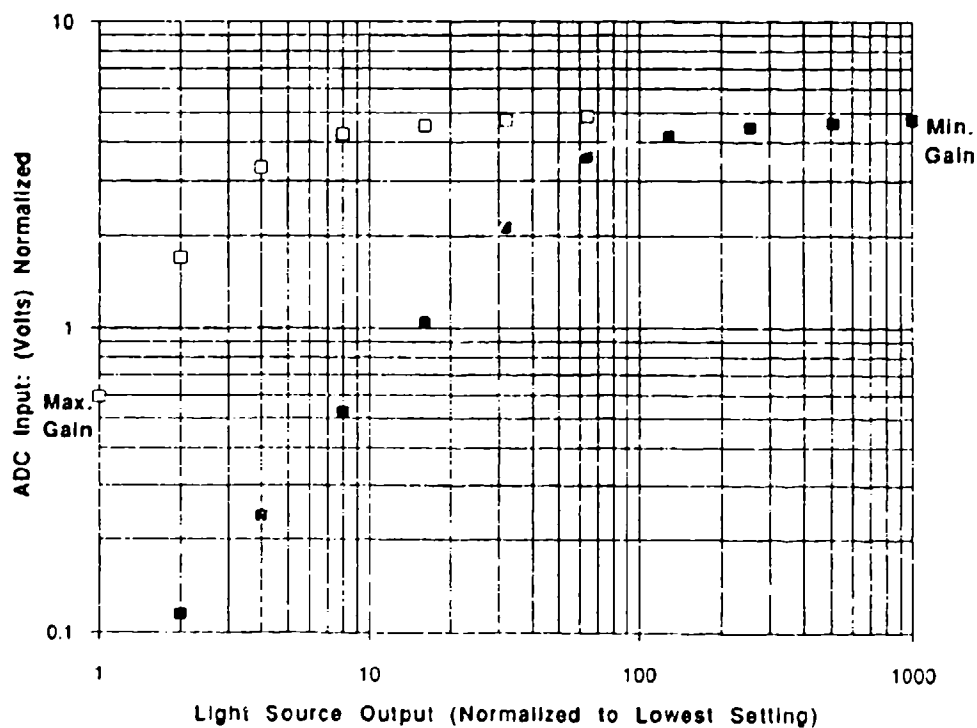
Figure 2.17b -- PhotoDiode Output: 68HC11 ADC Input
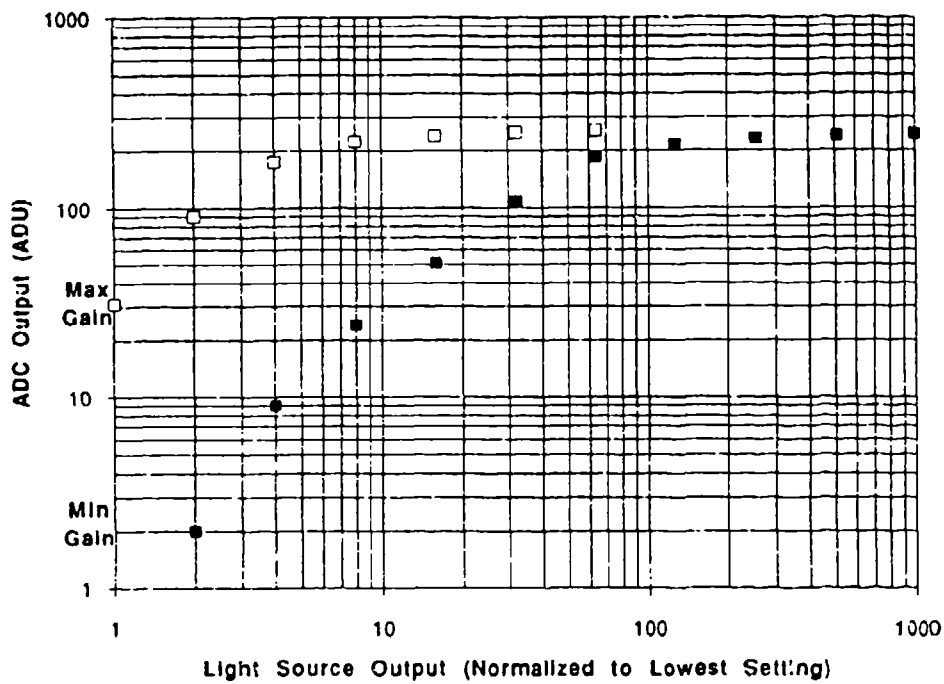
Figure 2.17c -- PhotoDiode Output -- ADC Ouput

PhotoDiode Output: PE4 .vs. ADU -- HAARP 8/93
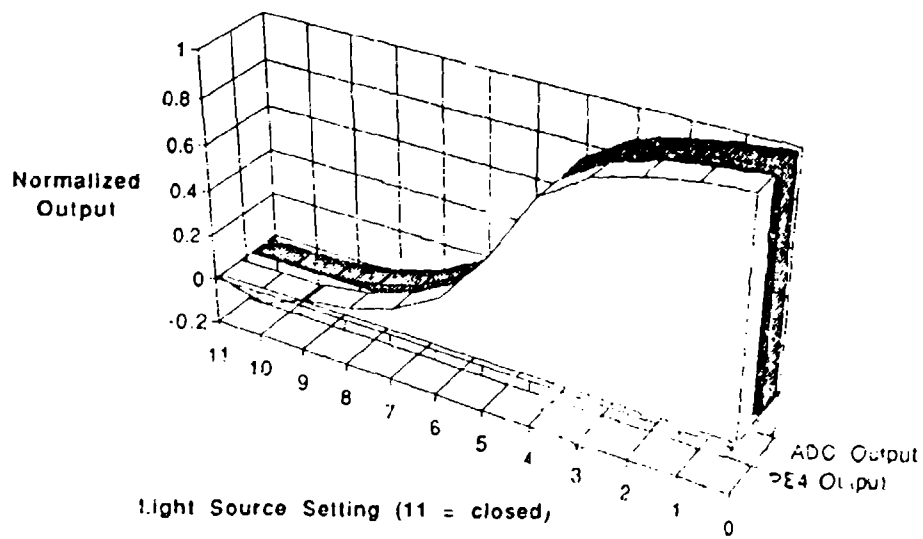
Figure 2.17d -- PhotoDiode Ouput -- ADC Linearity

2.5 Image Intensifier Performance

The HAARP imager has a 25mm Gen-2 Inverter intensifier with an S20ER photocathode and a P-20 phosphor.

2.5.1 VARO Specifications

```
SERIAL NO.  9301115

25MM 2ND GEN INVERTER WITH P-20 PHOSPHOR

PART # 510-3697-302

WHITE LIGHT PR @ 2854°K:   295 µA/lumen

PHOTOCATHODE SENSITIVITY:

        WAVELENGTH          PHOTORESPONSE
          (nm)                (ma/watt)

          900                    1
          830                   12
          800                   17
          700                   28
          600                   41
          550                   47
          500                   50
          450                   51
          400                   66
          350                   69
```

EBI (Equivalent Background Input) $1.6 \times 10\text{-}11$ (lumens/cm^2)

__LUMINANCE GAIN__

@ 5 X 10^{-6} INPUT ILLUMINATION: 90,000
INPUT CURRENT   16 milliamps

@ 5 X 10^{-5} INPUT ILLUMINATION: 6,665
INPUT CURRENT   15 milliamps

__OUTPUT BRIGHTNESS__

@ 5 X 10^{-4} INPUT ILLUMINATION: 1.2   F.L.
INPUT CURRENT   14 milliamps

@ 5 X 10^{-3} INPUT ILLUMINATION: 1.2   F.L.
INPUT CURRENT   14 milliamps

@ 5 X 10^{-2} INPUT ILLUMINATION: 1.4   F.L.

@ 10 X 10^{0} INPUT ILLUMINATION: OFF

CATHODE AND SCREEN QUALITY: OK

CENTER RESOLUTION: 32 LP/MM

Figure 2.18 -- VARO Intensifier Specifications

2.5.2 P20 Phosphor Spectral Curve

COLOR	C.I.E. COORDINATES	
	X	Y
Yellow-Green	0.426	0.546
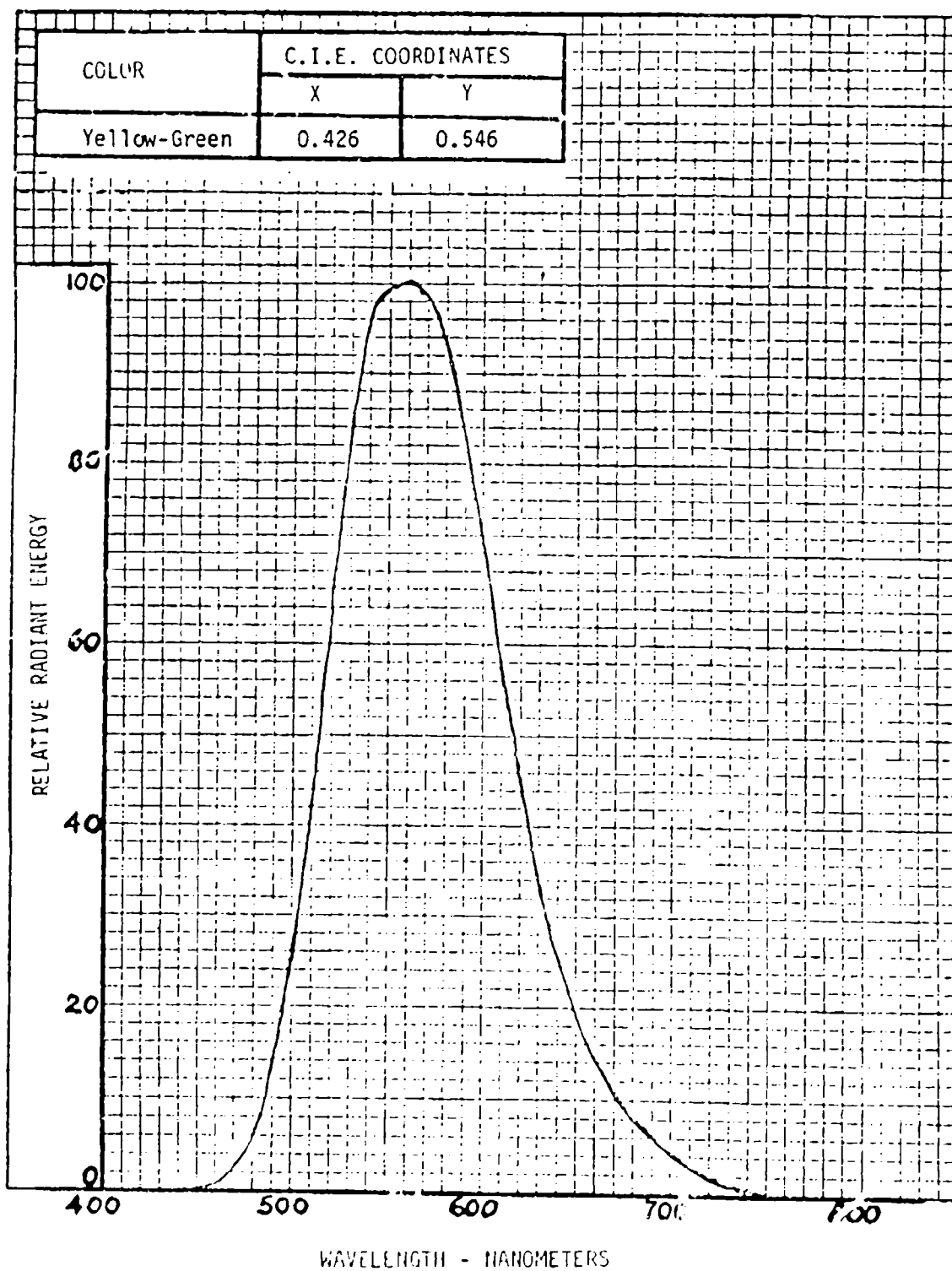
Figure 2.19 -- P20 Phosphor Spectral Output

2.5.3 Image Intensifier Resolution and CCD Resolution

The Image Intensifier is quoted from VARO as having a center resolution of 32 lp/mm at a Modulation Transfer Function of 2.5% which is what the eye can just detect. VARO quotes their tubes at three diffierent resolutions:

2.5 lp/mm	MTF: 90%
7.5 lp/mm	MTF: 60%
15 lp/mm	MTF: 25%

KEO projected a TV pattern directly onto the front of the image intensifier and looked at the intensifier resolution (**Figure 2.20**). To do this, the curvature correction lens and the cooler window were remo::ed.

As discussed in Section 2.1.4, the theoretical limit of the CCD at full resolution is 25 lp/mm. Because we are mapping a 10mm CCD image from a 25mm Intensifier image (100mm/42mm gives a 2.4:1 image reduction), this limit gets transformed back to 10 lp/mm on the image intensifier. According to the above specifications, we get an MTF of about 40% at this resolution.

This points out that the CCD is the limiting factor for the imager. Further, if we acquire images at 2x2 binning giving the CCD a pixel size of 40um, the CCD resolution limit is 12 lp/mm which corresponds to 5 lp/mm on the intensifier giving an MTF of about 75%. The resolution limit of the intensifier is quoted in the VARO specs at 32 lp/mm which would give 80 lp/mm resolution at the CCD (requiring 6um pixels corresponding to a 1632x1632 10.2mm^2 CCD!).

The real resolution of the imager is worse than this, however, because these calculations assume that the 'pixels' of the intensifier exactly overlap the pixels of the CCD. In practice, this never happens as light from on·· intensifier 'pixel' falls into adjacent CCD pixels thus lowering the effective MTF of the image.

43
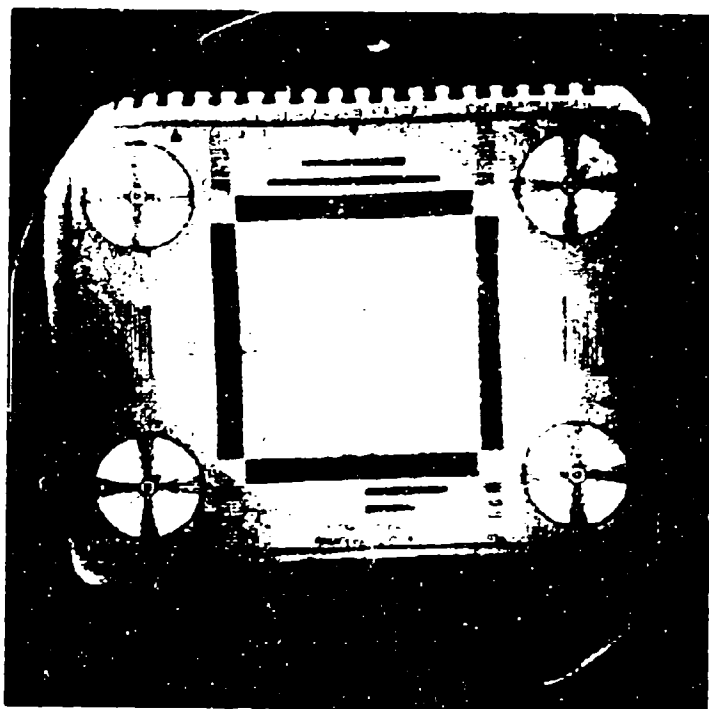
Figure 2.20 -- System Resolution with Intensifier

From examining the resolution image in **Figure 2.20**, one can by eye discern the four resolution lines down to about 200. Converting this to full resolution (accounting for fitting the rectangle in a circle, rather than the circle in the rectangle) corresponds to about 300 lines. 300 lines over the 25 mm intensifier comes out to about 12 lp/mm which is close to the theoretical limit of the CCD. One can see higher resolution by looking at the intensifier directly which confirms that the CCD is the limiting factor for the imager's resolution.

2.5.4 Image Intensifier Mask

On the resolution image in Figure **2.20**, one notices a bright ring around the edge of the intensifier. It was found during calibration that this ring is dependent on light at the edges photocathode. In Figure **2.21**, one can see this effect more dramatically.
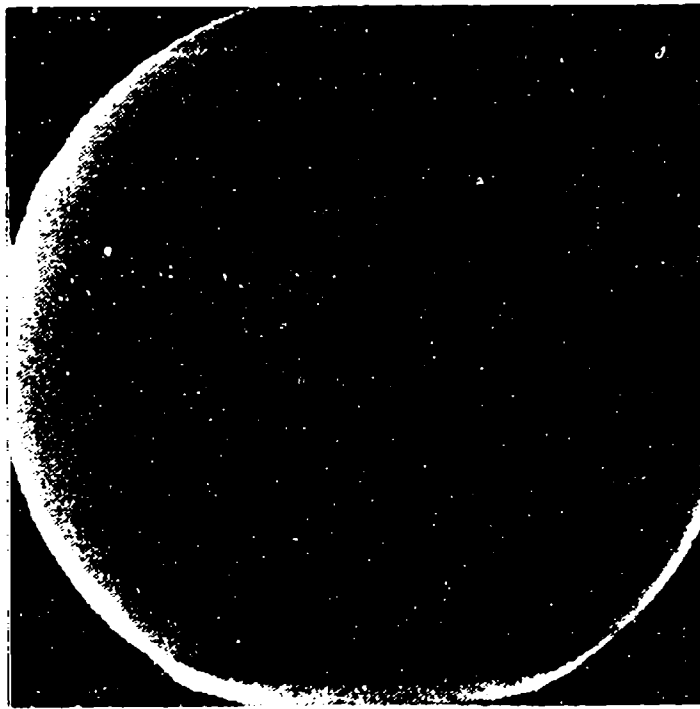
Figure 2.21 -- Image Intensifier Edge effects

KEO has noticed this problem frequently when using VARO intensifiers. An engineer at VARO confirmed that their tubes do have this problem and that it is related to the cleaning process of the micro-channel plates that increases the sensitivity at the edges of the image.

Since the actual image from the front optics is imaged to be 23mm in size on the photocathode of the intensifier it was decided by KEO to put a mask on the front of the intensifier that would block light from hitting the edges of the photocathode. The actual image size was first measured on the intensifier as shown in **Figure 2.22**

Measurements from Image

N = 473
Intens: 35 to 436
Image: 55 to 425
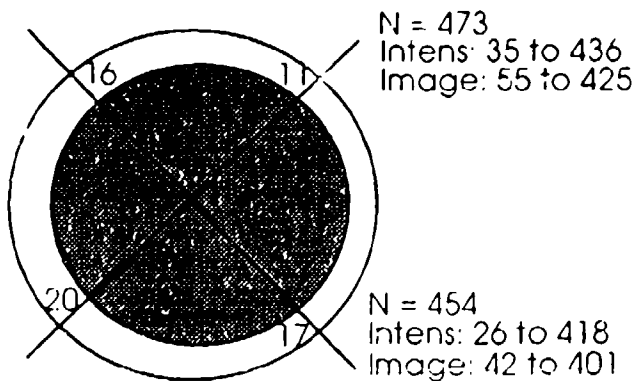
Front of the physical
Image Intensifier

N = 454
Intens: 26 to 418
Image: 42 to 401
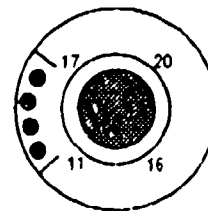
Figure 2.22 -- Fisheye Image on the Intensifier and it's measurement

45

As can be seen in the above figure, the orientation of the image in the intensifier was carefully measured and found to be slightly askew. The diameter of the intensifier in plot-points along the above plots is about 400 points or 16 points/mm. From the above measurements, the image is about 23 mm (image diameter: 370 points) which confirms the optics design. It should be noted for Figure 2.22 and the above numbers, that the term 'points' in this case is a relative diagonal of the CCD pixel size. This comes about from the way the AFG software reads a line of pixels at an angle. Assuming the plots were at about 45°, the CCD pixel dimensions would be about 1.4 times these values.

An Image Mask was fabricated out of Delrin with a 23.5mm ID offset .010" from the center of the mask assemby. This assembly was inserted into the recessed space of the curvature correction lens holder, between the lens and the intensifier. Care was taken to assure that the mask offset matched that of the image. An image was taken with no lens on the front (MASKNOLN.IMG) and with the Fish-Eye lens (MASKFSH2.IMG) to check that the mask was not blocking out any of the image. 10 pixel locations were measured along the edge of each of the circles and then the respective circle geometries were calculated in CCD pixels from these points:

Intensifier Image:	$R = 262$ pixels	$X_{org} = 246$	$Y_{org} = 267$
Intensifier Mask:	$R = 248$ pixels	$X_{org} = 244$	$Y_{org} = 266$
Fisheye Image:	$R = 243$ pixels	$X_{org} = 253$	$Y_{org} = 261$

The mask distorts slightly when placed in the lens holder and thus does not form a perfect circle in the intensifier image. As seen in **Figure 2.23**, the mask greatly reduces the edge affect of the image intensifier.

This image also points out the image centering problem talked about in Section 2.1.8 and fixed later during the calibration. In addition to centering the image in the CCD, the image needed to be centered somewhat in the intensifier image. To do this, the bayonet mount for the Canon 85mm lens was tightened with pressure applied towards the top of the intensifier housing, and a .005" shim was put under the front shutter to raise the Close-up lens slightly. As can be seen from the above data, the Fisheye image is now fairly well centered with respect to the CCD (dx = -2 pixels, dy = +6 pixels).

Using the data above, the 25mm intensifier has a radius of 262 pixels, or 10.5 pixels/mm. This gives an image size of 243/10.5 = 23.1 mm which confirms the optics design. This also shows that the image is offset from the CCD's optical axis by -0.2mm (X) and +0.6mm (Y).

Figure 2.23 -- Image Intensifier with Mask (no front lens)

2.5.5 Image Intensifier Flat-Field -- System Flat-field

Getting an accurate flat-field image of the system has always been a very challenging task for the calibration of the instrument. Assuming all non-uniformities in front of the image intensifier have spherical symmetry (vignetting curves), a good flat-field of the system from the image intensifier back will be fairly accurate. Of course, the non-uniformity of the filters are not taken into account in this analysis. One could isolate the non-uniformity of the intensifier by taking into account the 7% vignetting of the re-imaging optics (Section 2.2) and using the CCD flat-field image (Section 2.1.3).

VARO Image Intensifier tubes are not rated to be very uniform as they are produced for a mil-spec standard where uniformity is not a necessary requirement. The factory specification for uniformity is to within 2:1. Visual inspection of the HAARP image intensifier revealed an unusually uniform tube compared to most of the tubes KEO Consultants has received in the past.

To get an intensifier flat-field, the curvature correction lens had to be removed. **Figure 2.24** demonstrates the calibration setup used. To cut down on ambient room light distorting this flat-field, the calibration was done at night with great care taken to minimize contaminating light sources.
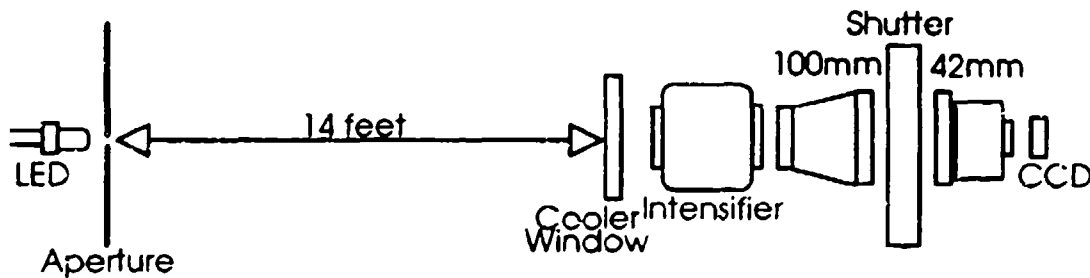
Figure 2.24 -- Image Intensifier flat-field Setup

The intensifier tube used in the HAARP imager was delivered with a back fiber-optics plate mis-alignment of greater than .005" (.002" out of specification). Because of this, there is a very slight focusing discrepancy between the different sides of the intensifier. The re-imaging optics were focused to compromise between the difference.

Three images were taken. (1) Intensifier Dark Noise (2) Room integration [LED turned off] (3) LED integration. Each image is an addition of 5 images to get better statistics. Examination of the Room integration and the LED integration shows that the ambient light in the room accounted for about 4% of the light in the LED integration. This is important as the ambient light in the room is not guaranteed to be uniform over the photocathode of the image intensifier. Profiles for these two images are given in **Figure 2.25**.

Examination of this flat-field image, shows a maximum non-uniformity of around 1.6:1 which confirms our visual observation of this being an unsually uniform tube (by VARO's specifications). This image (**Figure 2.26**) also clearly shows the results of the image intensifier mask (Section 2.5.4) and the results of the CCD orientation (Section 2.1.8).

48
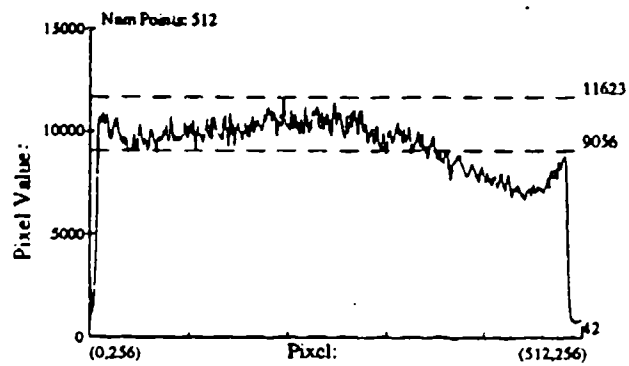
Figure 2.25a -- Profile of Intensifier Flat-Field image
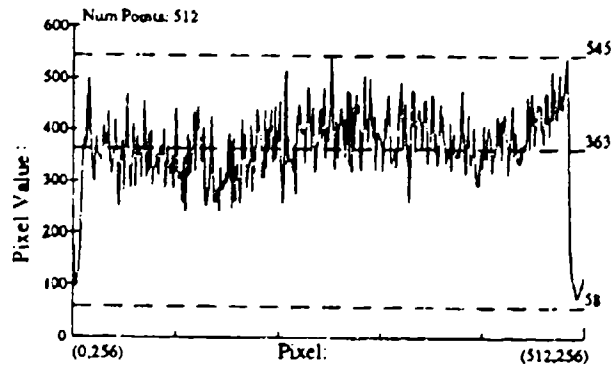
Figure 2.25b -- Profile for Ambient Light in Flat-Field
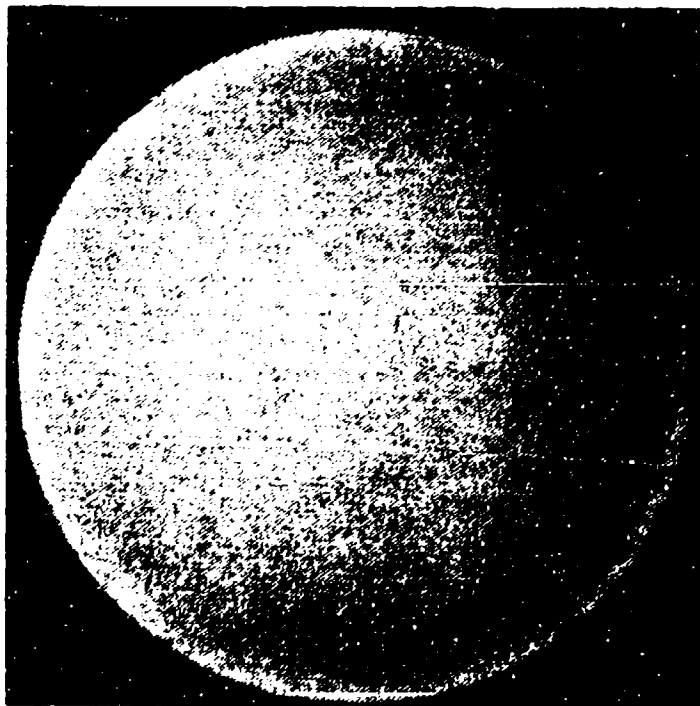
Figure 2.26 -- Image Intensifier Flat-Field Image
[Stretched: Black = 200, White = 17000]

49

2.5.6 Image Intensifier Gain Calibrations

There are four gains for the image intensifier that can be set either via the instrument's control panel, or by computer control. These gains are notated as 0, 1, 2, and 3 going from minimum gain to maximum gain. A calibrated light source was used to adjust the intensifier gains by adjusting potentiometers VR5-VR7 on the KEO Interface board (see Hardware manual). The minimum and maximum gains are determined by the maximum dynamic range of the image intensifier and is typically in the neighborhood of 10. The results of the 8/27/93 gain calibration are:

Gain	Exposure	Signal-Noise	Normalized	Ratio(min)	Ratio(max)
0	10 sec	3235	3235	1	0.07
1	10 sec	6593	6593	2.04	0.14
2	5 sec	6475	12950	4.00	0.28
3	2.5 sec	11698	46792	14.46	1

We notice from this calibration that this particular intensifier tube has an unusually high gain range. This ratio can also be seen on the Photodiode calibration plots in Section 2.4. Since the low gain settings are used more often (to reduce intensifier noise), the first three gains were calibrated to be ratios of 2.

2.5.7 Intensifier Dark Noise

Similar to the CCD, the intensifier has dark current mostly associated with the Trialkali Photocathode. This dark current is temperature dependent as is displayed in **Figure 2.27**.
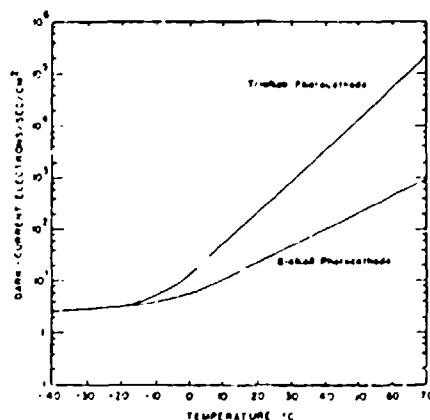
Figure 2.27 -- Photocathode Dark Current

There is also image intensifier noise that appears as much brighter spots, called "ion-noise". This is due to positive ions created at the input of the micro-channel plates. There is an aluminized layer on the front of the MCP which is designed to stop these ions from drifting back to the cathode. Occasional ions are accelerated back into the photocathode, emitting a burst of electrons, creating a large flash of light. When looking at the output of an image intensifier, it is easy to distinguish between the two types of noises. In calculating the average dark current for the image intensifier, these occasional "ion" spots were avoided.

Curves for the intensifier dark current are shown in **Figure 2.28**. Notice that, similiar to the CCD, we get about a factor of 2 reduction in dark current for every 6°C of cooling (or factor of 10 for every 20°C of cooling). It can be seen from **Figure 2.27** above, that only about another factor of 10 could be achieved by cooling the photocathode down to about -20°C. The HAARP intensifier designed was chosen to cool to around 1->5°C to avoid condensation on the cooler window. Typically, the cooler achieves a differential of about 20°C.
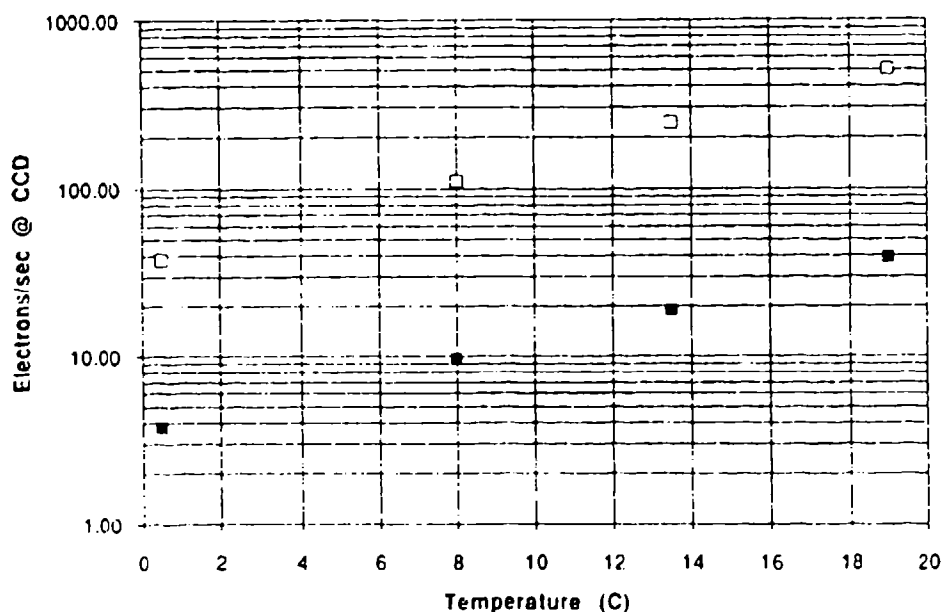
Figure 2.28 -- HAARP Intensifier Dark Current

The intensifier dark current .vs. temperature was measured by disconnecting the Intensifier Cooler TEC power and putting different valued resistors in series to reduce the effective cooling power of the TEC. The CCD was still cooled to it's maximum during these exposures and the CCD dark current was subtracted out to get only the contribution of the image intensifier photocathode.

2.6 Filters

Filter transmission curves were measured for the HAARP imager and are as follows:

Wavelength (λ)	% Transmission	Width (Å)
4282	61.5	21.9
4867	72.4	28.6
5300	68.7	24.7
5581	76.4	15.8
6304	77.3	15.6
7778	70.8	15.6

The Filter curves a shown in Figures 2.30 through 2.35.

A flat-field for the filters was not measured for this calibration due to the difficulty of attaining an accurate measurement. In order to measure this flat-field, we would need a uniform, monochromatic light source. Optically, to get a telecentric uniform image projected on the filters, we would need to build a setup shown in Figure 2.29.
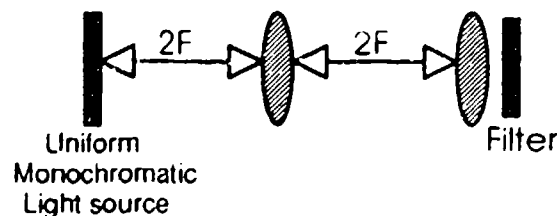
Figure 2.29 -- Optical Setup for flat-fielding Filters

Figure 2.30 -- Transmission curve for 4282Å Filter
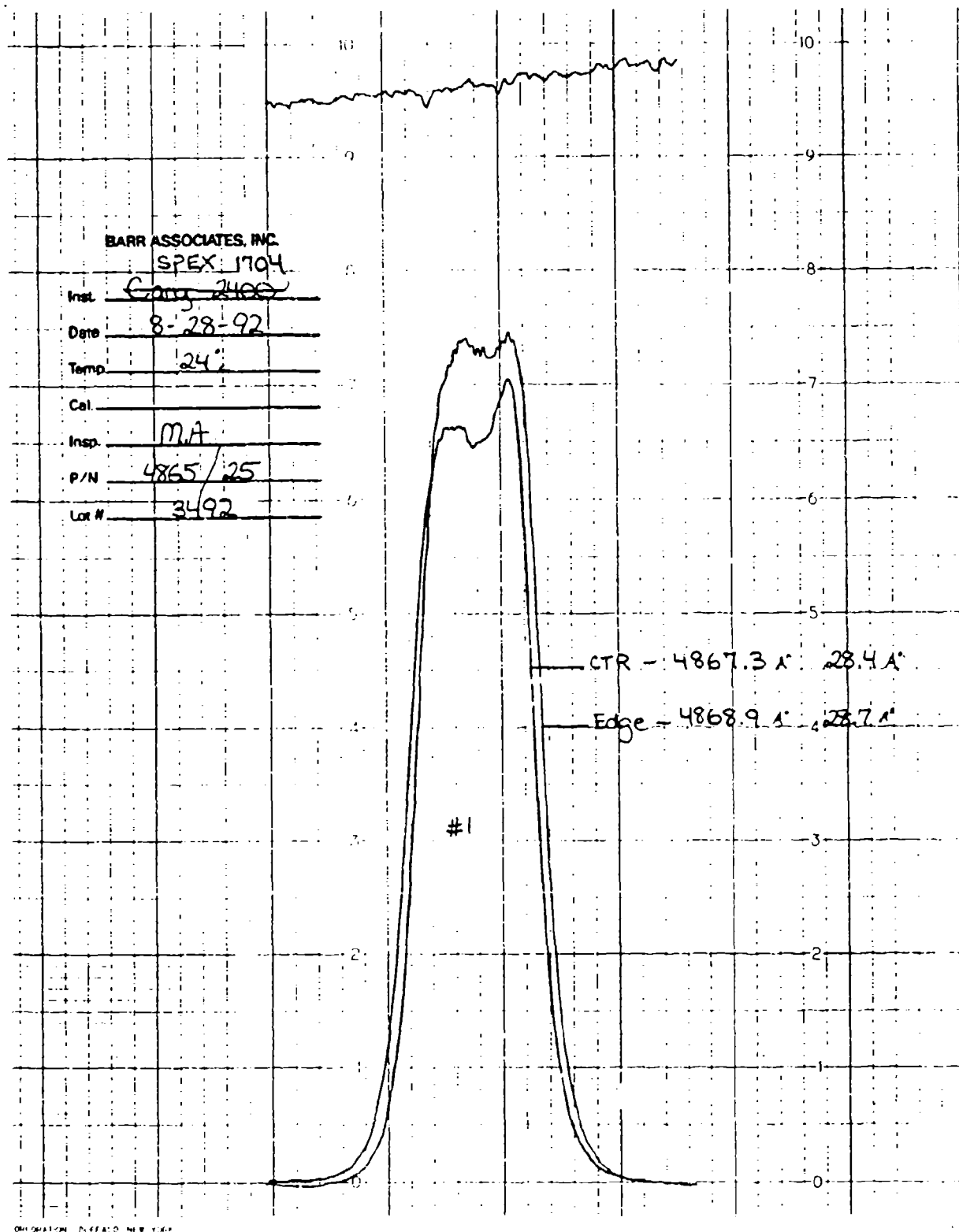
Figure 2.31 -- Transmission curve for 4867Å Filter

BARR ASSOCIATES, INC.

Inst. SPEX 1704

Date 7-28-92

Temp. 7.24°

Cal.

Insp. INA

P/N 5300/25

Lot # 3092

	CWL	BW
CTR	5300.2 A°	24.7 A°
Edge	5301.9 A°	24.7 A°

#1
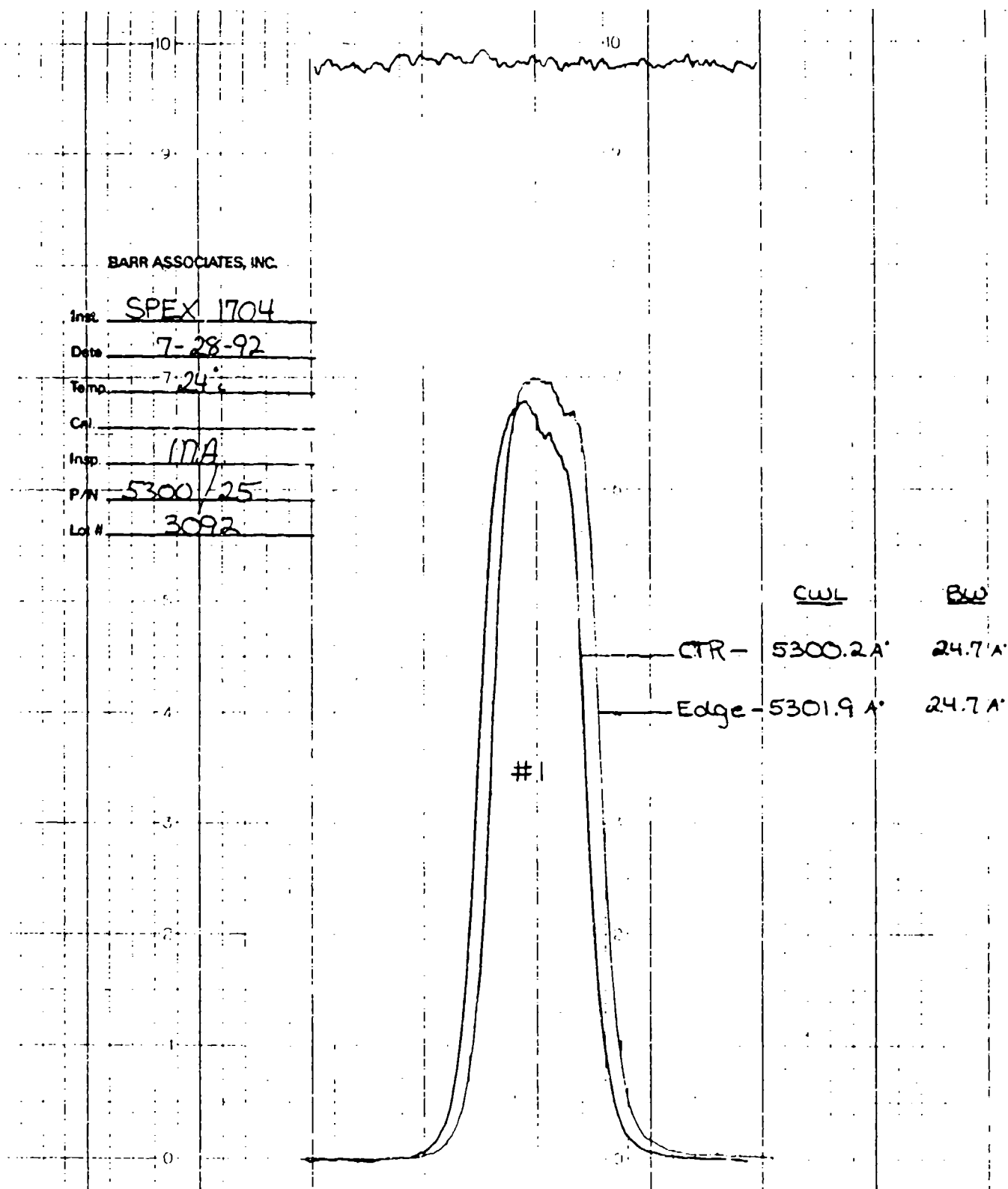
Figure 2.32 -- Transmission curve for 5300Å Filter

BARR ASSOCIATES, INC.

Inst.	SPEX 1704
Date	7-31-92
Temp	24°
Cal.	
Insp.	M.A
P/N	5581/15
Lot #	3092

CTR - 5582.5 nm 16.0 nm

Edge - 5583.8 nm 15.7 nm
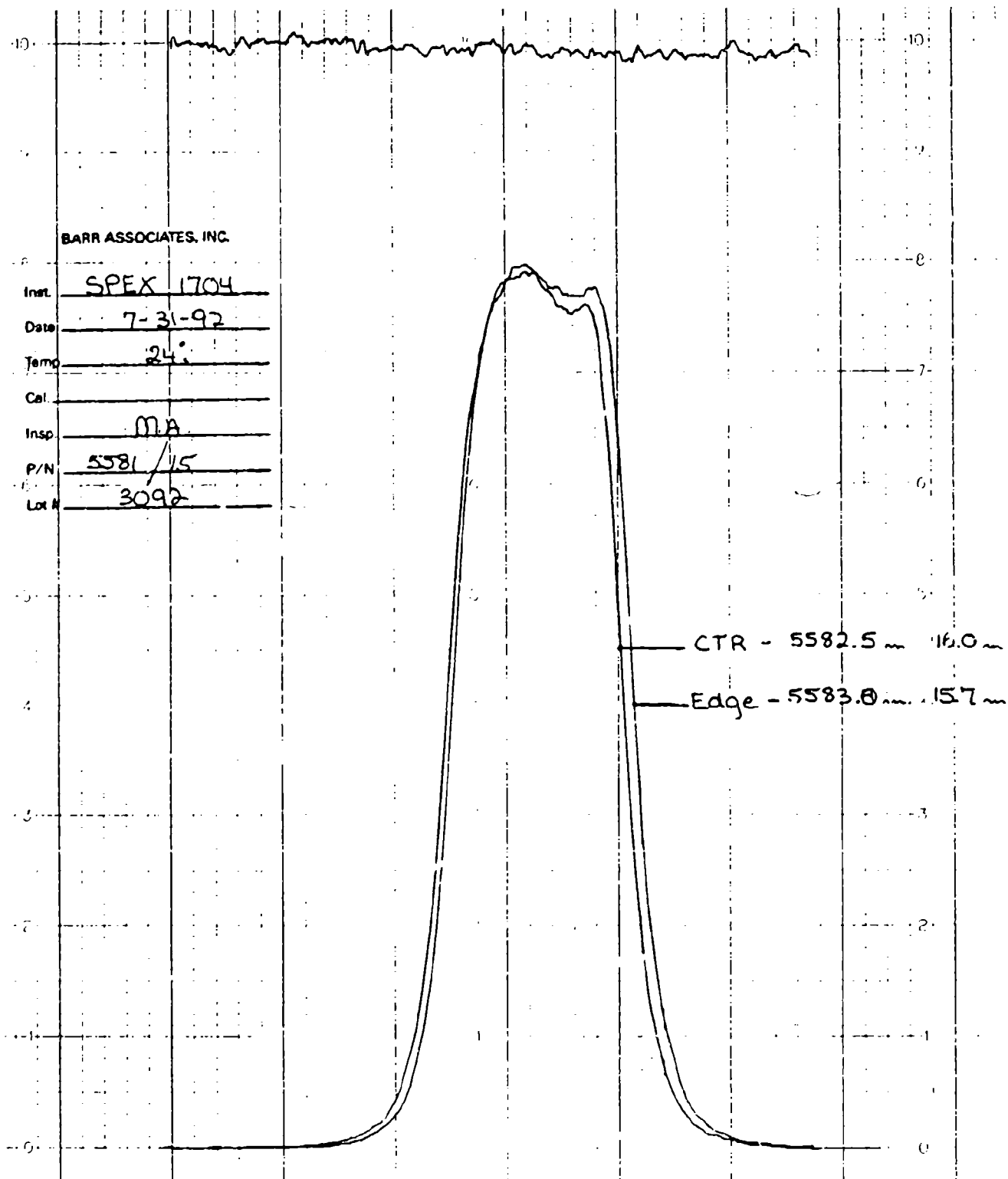
Figure 2.33 -- Transmission curve for 5581Å Filter

BARR ASSOCIATES, INC.

Inst. SPEX 1704

Date 8-6-92

Temp. 24°c

Ca.

Insp. M.A.

P/N 6304/15 Å°

Lot # 3192

	CWL	BW
Edge—	6304.8 Å	15.5 Å
Ctr.—	6304.4 Å	15.7 Å

Figure 2.34 -- Transmission curve for 6304Å Filter

57
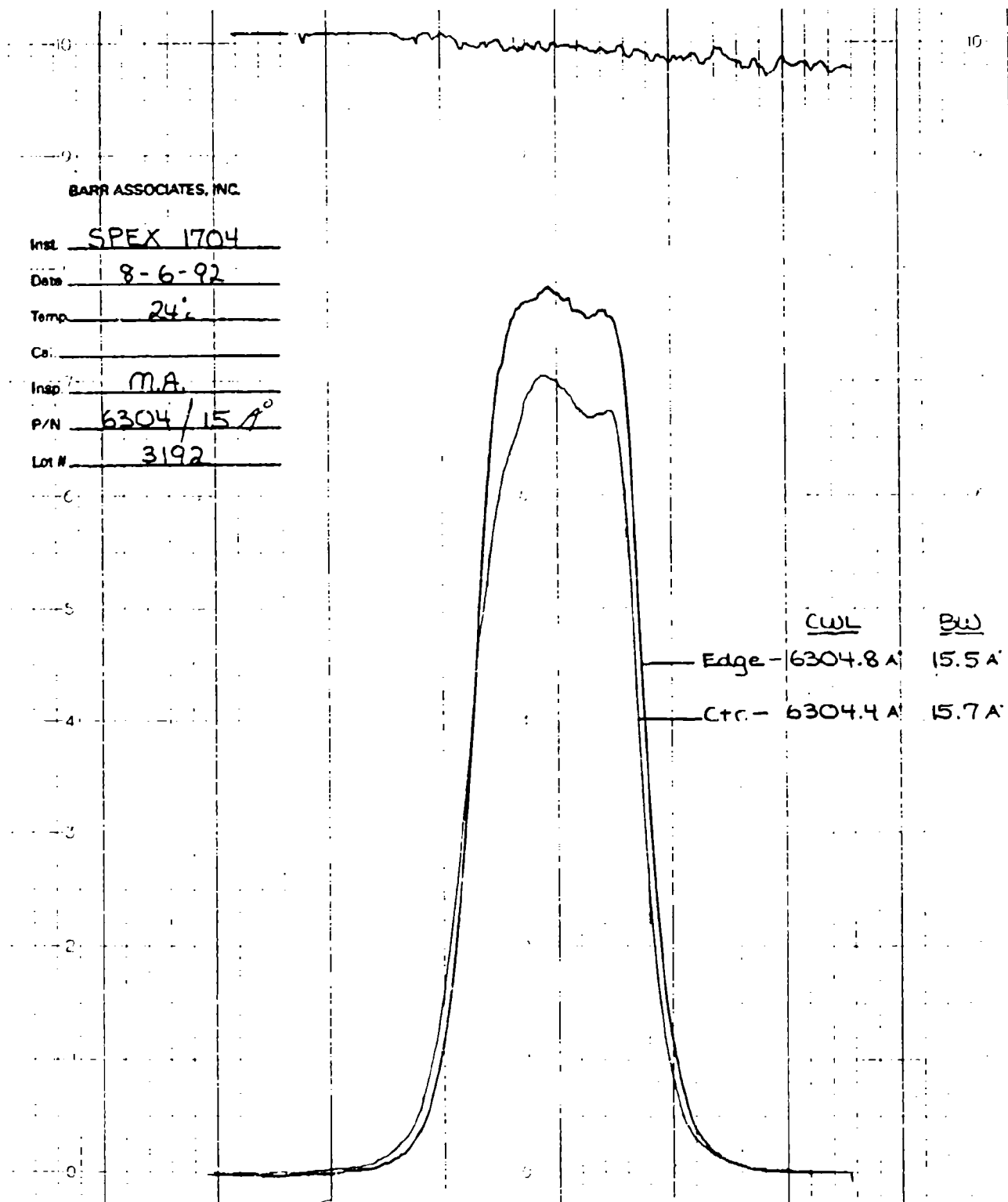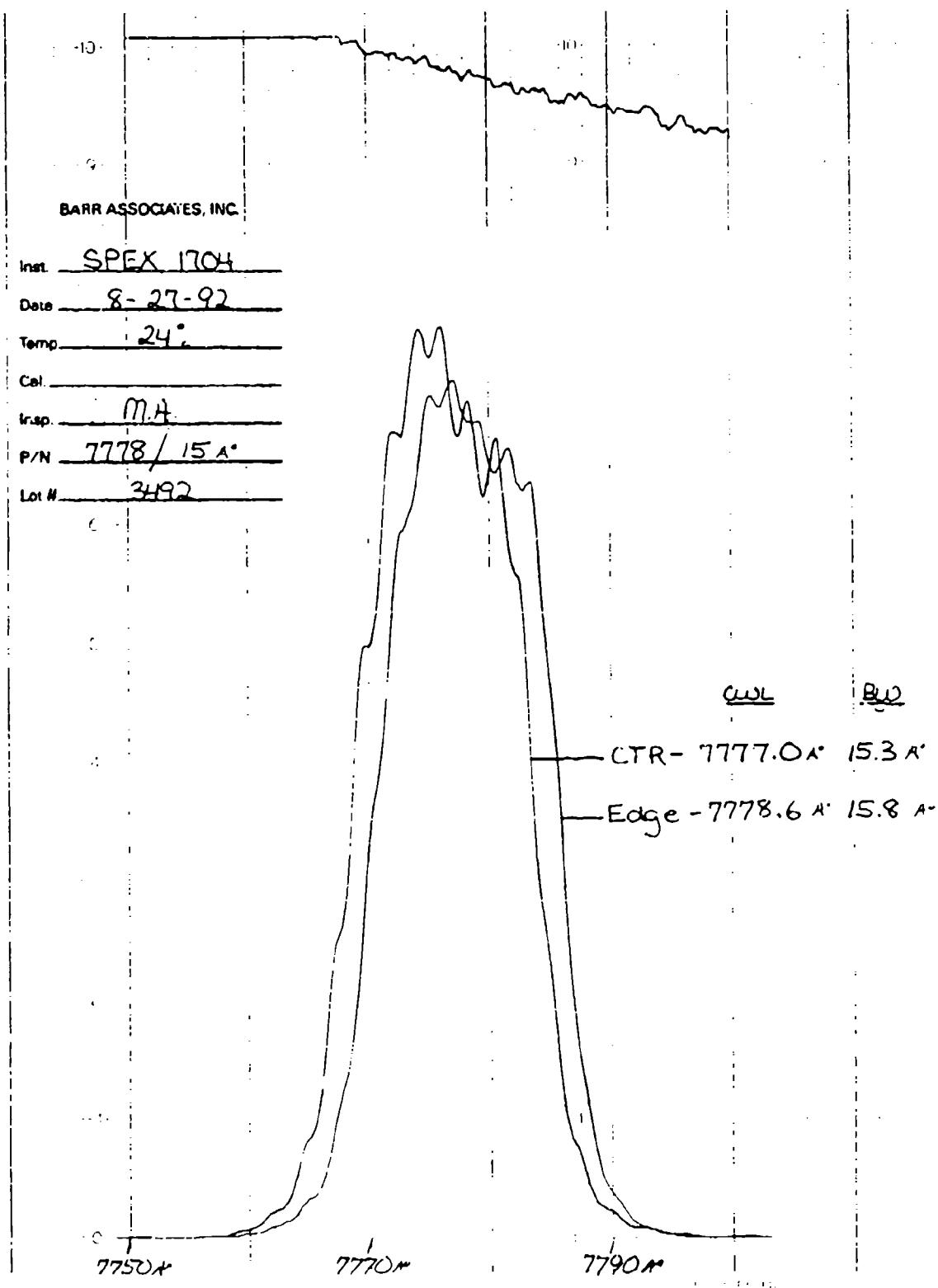
Figure 2.35 -- Transmission curve for 7778Å Filter

The interference filters require an image with a cone angle of 7° (or $f4$). Two 4" achromat lenses could be placed a distance of two focal planes apart:

$$\text{For } f4, \text{ and d} = 4\text{":} \quad F = f \times d = 16\text{"}$$

Thus, a 32" tube could be build with the two achromats at either end of the tube to get a valid flat-field image at the filter using a 4" uniform lights source. By adjusting the distances between the light-source and the lenses, we could reduce the size required for the monochromatic light-source.

A monochromatic light-source is needed because the filter transmission has a spatial variation across the filter. Thus, for a 5577Å line the transmission will vary from point to point in the filter. To measure this non-uniformity would require very narrow-band filters at the light source. Such a calibration would be very expensive, and was not attempted.

2.7 Spectral Calibration

The HAARP Imager was calibrated for spectral sensitivity using KEO's Light Source #2. The spectral output curve for the light source is given in Figure 2.36.

KEO Light Source #2   [870mA Ou1u]
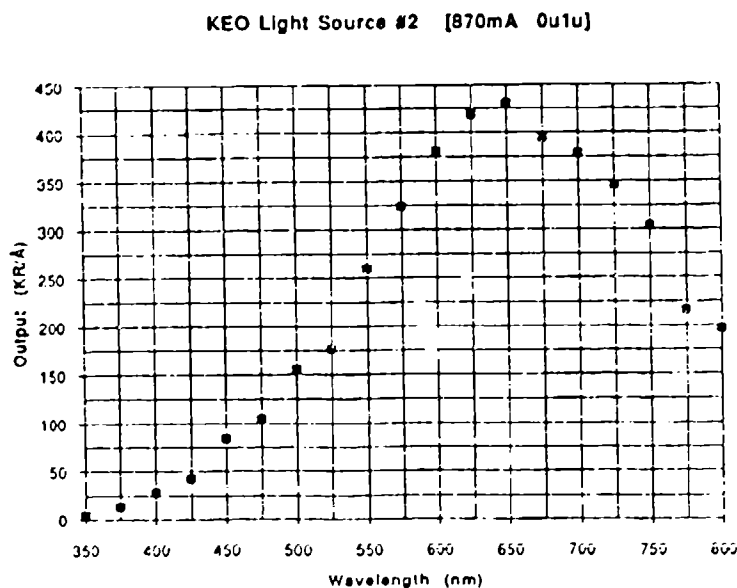
Figure 2.36 -- Spectral Output for KEO Ligh. Source #2

59

To calibrate the instrument, the light source was placed in front of the instrument in a dark room. For each filter, five images were taken and added together using a 5-10 second exposure, intensifier gain at 1, CCD gain at LO and 1x1 binning. The light-source setting was 0d6d.

Wavelength	Image	Background	Signal x (10/Exp)
4282Å	3207	153	3054
4867Å	13584	153	17908
5300Å	15039	134	29810
5581Å	13905	133	27544
6304Å	3278	99	22284

(These were taken MID gain, 4 second exposure)

Wavelength	Image	Background	Signal x (10/Exp)
6304Å	14291	53	14238
7778Å	2585	53	2532

Multiplying the filter transmission and width times the light source's output, we get the following results:

Wavelength	Transm(%)	Width(Å)	Output(R/Å)	Signal (R)
4282Å	61.5	21.9	6.83	92.0
4867Å	72.4	28.6	18.53	383.7
5300Å	68.7	24.7	27.98	474.8
5581Å	76.4	15.8	40.60	490.1
6304Å	77.3	15.6	59.91	722.4
7778Å	70.8	15.6	30.73	339.4

Normalizing the light source outputs to 6304Å, we get:

Wavelength	Signal	Relative Sensitivity
4282Å	0.14	1.10
4867Å	0.80	1.51
5300Å	1.34	2.04
5581Å	1.24	1.83
6304Å	1.00	1.00
7778Å	0.18	0.38
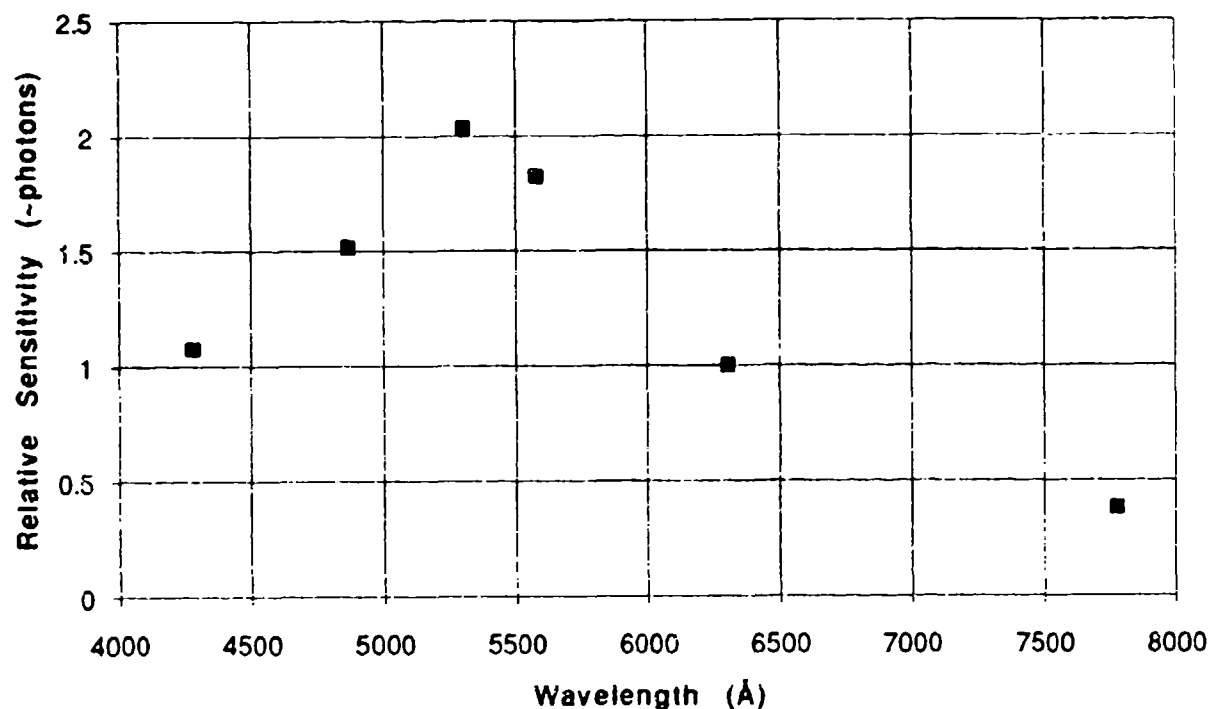
Figure 2.37 -- HAARP Imager Spectral Sensitivity

This sensitivity is a function of the image intensifier photocathode response and closely matches the curve for our S20ER photocathode. It should be noted that commercial response curves are usually presented in units of mA/Watt instead of photons. The above data can be scaled to these units by dividing the relative sensitivities by the wavelength. **Figure 2.37** show the Imager's spectral sensitivity normalized to 6300Å in terms of photons detected at the CCD.

2.8 Temperature Calibration

The temperature monitoring circuits were calibrated for the HAARP imager 4/26/93. A detailed explanation of these circuits is given in Section 3.4.1. The results of the calibration are:

Set: T_{REF} = 4.01 V V_{DISPL} = 1.956 V i_{RTD} = 2.00 MA

CCD Measurements:

V_{CCD}:	-4.41V + 4.01V = -0.4V
PE0:	.868V
T_{DISPL}:	-27.8C

G_{347}:	-2.17
ADU_{CALC}:	44 --> -28C
RTD_{TBL}:	89.2Ω x 2 mA = .1784V
$G_{101-CALC}$:	24.7

T_{COMP}: -27.5C	
ADU_{CALC}:	45

TEC Measurements:

V_{TEC}:	-5.72V + 4.01V = -1.71V
PE1:	3.49V
T_{DISPL}:	-39.4C

G_{347}:	-2.04
ADU_{CALC}:	179 --> 39.5C
RTD_{TBL}:	115.3Ω x 2 mA = .1784V
$G_{101-CALC}$:	24.8

T_{COMP}: 41.0C	
ADU_{CALC}:	182

INT Measurements:

V_{INT}:	-4.85 + 4.01V = -0.84V
PE0:	1.714V
T_{DISPL}:	-6.1C

G_{347}:	-2.04
ADU_{CALC}:	88 --> -6C
RTD_{TBL}:	97.65Ω x 2 mA = .1953V
$G_{101-CALC}$:	24.8

T_{COMP}: -5.5C	
ADU_{CALC}:	87

FILT Measurements:

V_{FILT}:	-5.41V + 4.01V = -1.40V
PE0:	2.804V
T_{DISPL}:	21.7C

G_{347}:	-2.00
ADU_{CALC}:	144 --> -22C
RTD_{TBL}:	108.5Ω x 2 mA = .217V
$G_{101-CALC}$:	24.9

T_{COMP}: 23.0C	
ADU_{CALC}:	146

T_{REX-10}:	70.6F --> 21.4C

	Intensifier		**Filter Wheel**
RTD_{MEAS}:	.1939 V / 2 mA = 96.95Ω	RTD_{MEAS}:	.2175 V / 2 mA = 108.75Ω
V_{101}:	4.85V	V_{101}:	5.45V
G_{101}:	25.01 (Actual Gain)	G_{101}:	25.06 (Actual Gain)

2.9 System Vignetting

As discussed in Section 2.5.5, a system flat-field was taken that accounts for all system non-uniformities and lens vignetting from the intensifier back to the CCD. The major source of vignetting for the imager is the front primary lenses. This vignetting is intrinsic to the lens design of these commercial lenses. The HAARP imager has two lenses: the fisheye and the 300mm lens. System vignetting with both these lenses was measured at maximum aperture by setting up a small light source and then scanning this source across the field-of-view of the image in both directions. This vignetting is intrinsic to the lens design of these commercial lenses.
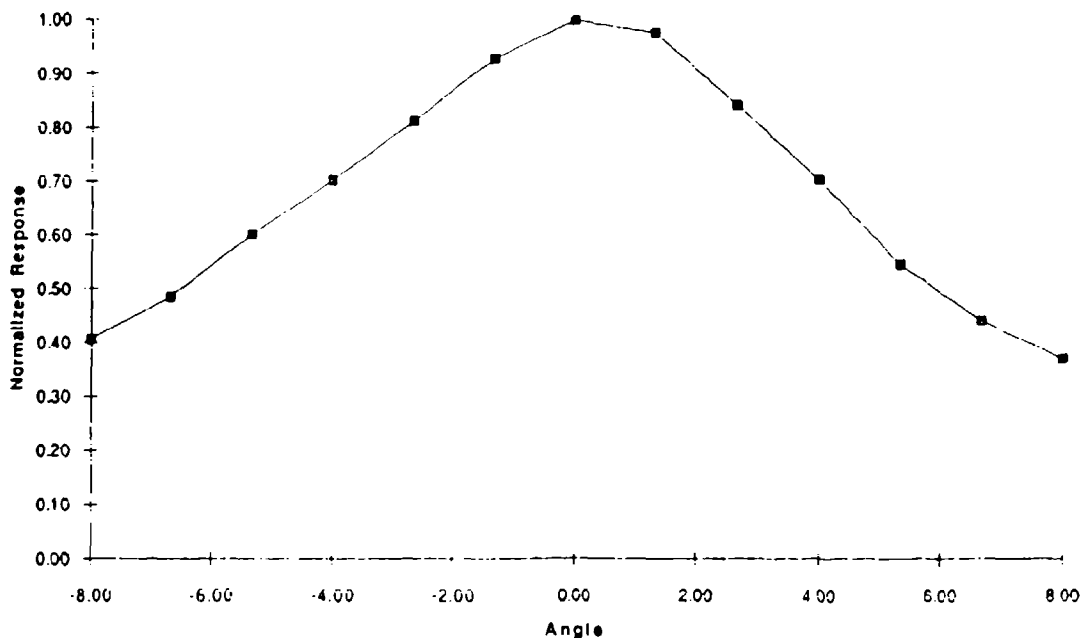
Figure 2.38 -- System Vignetting with 300mm Lens
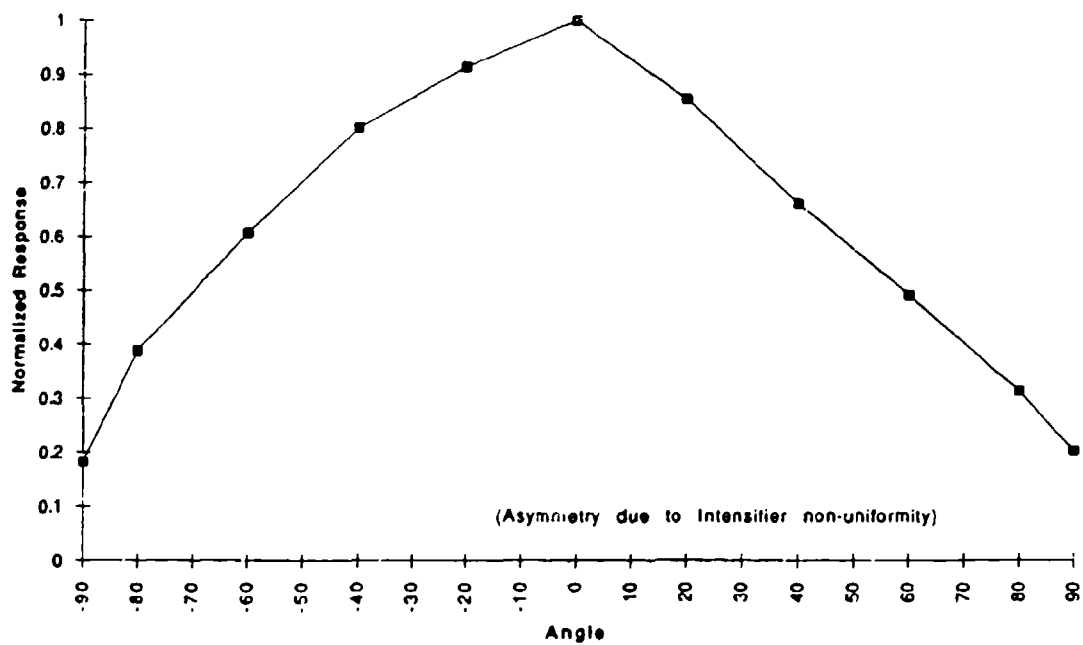
Figure 2.39 -- System Vignetting with Fisheye Lens

In addition the radial linearity of the fish-eye was measured:
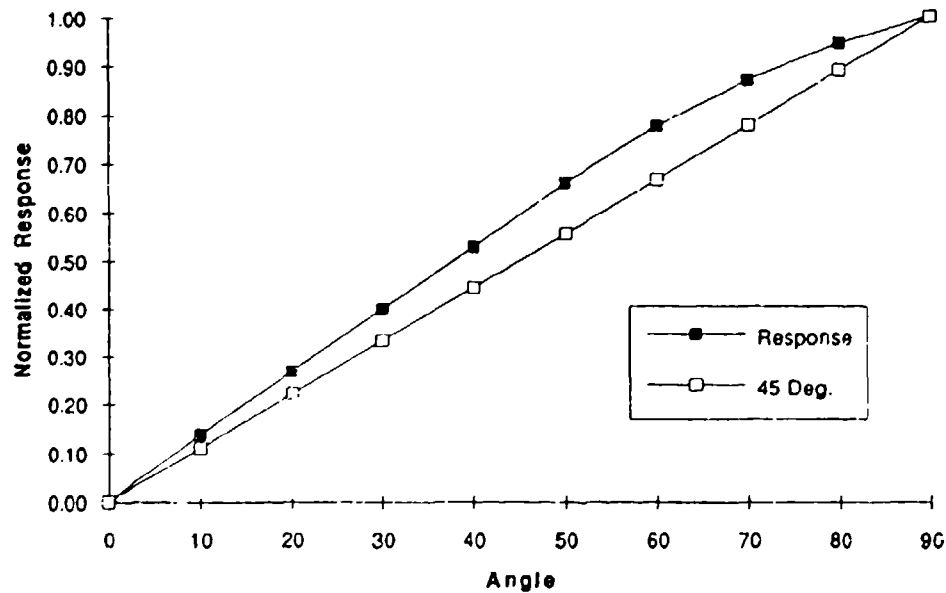
Figure 2.40 -- Radial Linearity of Fisheye lens

2.10 System Sensitivity

2.10.1 Theoretical case for a single photon

It is usefull to understand the system gain in terms of a theoretical photon being detected at the photocathode of the image intensifier. The image intensifier gain is quoted at 90,000. This specification refers to a steady stream of photons at 2584K (Tungsten). Since the quantum efficiency of the photocathode is on the order of 10%, this means that each detected photon, must create on the order of 900,000 photons at the output of the tube. Correcting this for the response of photons at 5577Å, we get about 180,000 photons for every detected photon (or a fifth the number of photons from a Tungsten source spectral distribution).

As noted in Section 2.2, the relay optics has an efficiency of around 10%, so there are only 18,000 photons arriving at the CCD. Taking into account the quantum efficiency of the CCD (~30%), we have about 5500 electrons collected by the CCD. Because these photons will be spread over one to four pixels, we can predict single photon events as the CCD for the different gain settings, and intensifier gain settings:

Int. Gain	Event (e-)	HI (ADU)	MID (ADU)	LO (ADU)
ADC Gain:		[25e-/ADU]	[49e-/ADU]	[91e-/ADU]
Read Noise:		[24e-]	[39e-]	[63e-]
MAX	1375-5500	55 - 220	28 - 112	15 - 60
MIN	95 - 380	3 - 15	1 - 8	1 - 4
MINx2	190 - 760	7 - 30	3 - 15	2 - 8
MINx4	380 - 1520	15 - 60	7 - 31	4 - 16

We see that the coupling between the intensifier and the CCD is probably marginal to resolve single photon events for MINIMUM intensifier gain. Thus, for extremely sensitive images trying to resolve single photon events, the minimum coupling that should be used is:

Intensifier: MIN gain & CCD: HI gain

This confirms that LO gain should be used to extend the dynamic range of the ADC conversion and used for higher light levels. It should be noted that the above numbers are estimates for photons created at 5577Å, and that there would be less photoelectrons from photons at other wavelengths (see **Figure 2.37**).

2.10.2 Theoretical Sensitivity of the HAARP Imager

The following is a simple and quick calculation of expected sensitivity of the HAARP system:

Each image pixel at the CCD translates to an area at the filter; the image diameter at the filter of 3.5"(89mm) covers 512 CCD pixels, so the pixel size at the filter is

$$89/512 = 0.174 \text{mm, giving a pixel area of } 3.0 \times 10^{-4} \text{ cm}^2$$

The F4.5 Pentax lens collects from a half-angle of 6.4° (0.112 radians), or a solid angle of

$$\pi \times [0.112]^2 = 3.9 \times 10^{-2} \text{ steradians}$$

1 Rayleigh (R) of source luminance corresponds to

$$10^6 / 4\pi = 7.96 \times 10^4 \text{ photon/cm}^2 \text{ sec ster}$$

If we assume a Quantum efficiency at the peak of the cathode spectral response of **12%**, overall lens transmission of **70%**, filter transmission of **70%**, and assume that every cathode photoelectron event is amplified enough by the image intensifier to give a detectable CCD response, then the expected count rate/pixel at the CCD for 1 R input is

$$3.0 \times 10^{-4} \times 3.9 \times 10^{-2} \times 7.96 \times 10^4 \times 0.12 \times 0.70 \times .70$$

$$= 5.5 \times 10^{-2} \text{ counts / pixel-sec}$$

Thus, for every pixel to have 1 event during a 1 sec exposure, the required source luminance is about 18R.

2.10.3 Sensitivity Measurements

To take a specific example from our calibration, we go through this measurement for 4867Å. At this wavelength, the photocathode (S20ER) efficiency is about 9%, the filter transmission is .724. We now have a minimum source luminance of about

$$= 4.2 \times 10^{-2} \text{ counts / pixel-sec}$$

which gives about 24R for 1 event/pixel-sec. Using the KEO light source, we illuminate the the imager with 384R using the 4867Å filter. Images were taken with the Nikon camera lens stopped down to $f11$, giving a source of ~5R. Thus from the above estimation, we should see photon events in about 20% of the pixels covered by the source. If we take an image at $f8$, we would have a source of ~9R and would expect about 36% of the pixels filled. An image at $f16$ would give ~2.2R giving us about 9% of the pixels filled.

Carefully, looking at these images and using histograms, we can see that these numbers roughly agree.
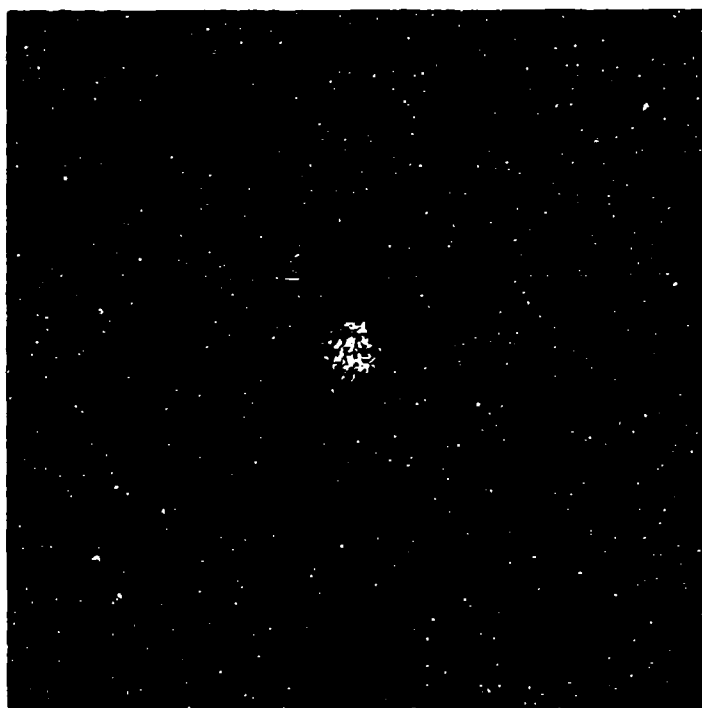
Figure 2.41 -- 4867Å filter, Int: 0, CCD: HI: Exp: 1 sec $f4$ 35R-sec
[Stretched: Black = 22, White = 45]

Figure 2.42 -- 4867Å filter, Int: 0, CCD: HI: **Exp: 1 sec f8 9R-sec**
[Stretched: Black = 23, White = 32]
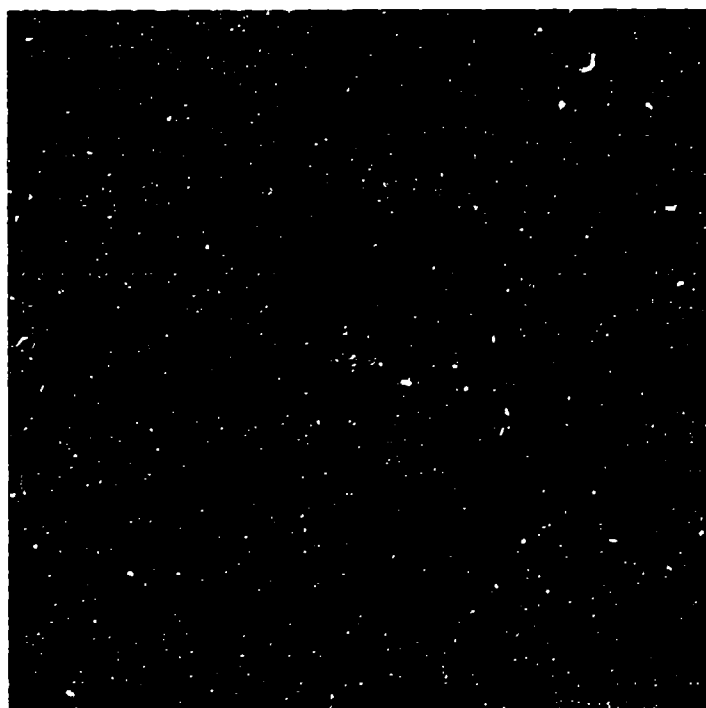
Figure 2.43 -- 4867Å filter, Int: 0, CCD: HI: **Exp: 1 sec f11 5R-sec**
[Stretched: Black = 23. White = 30]

Figure 2.41 shows the calibration setup for fairly good photon-statistics. A 35R source is integrated for 1 second giving a 35R-sec image. Note from Section 2.10.1, that a single 5577Å photon event with the coupling Int: 0 and CCD gain at HI will produce a signal of 3 - 15 ADU depending on how many pixels the resultant photons cover.

Figure 2.42 shows the sensitivity of the instrument at 9R-sec using a 1 second exposure. Enlarging the center of the image and carefully counting the pixels that have received some photons, reveals a percentage of pixels filled on the order of what was predicted for 9R-sec in Section 2.10.2.

Figure 2.43 shows the sensitivity of the instrument for 5R-sec using a 1 second exposure. This image again confirms the results of Section 2.10.2. Note that the intensity of each photon-event is the same as **Figure 2.42**, except that there are about 1/2 as many such events in the area of the source (on the order of ~20%).
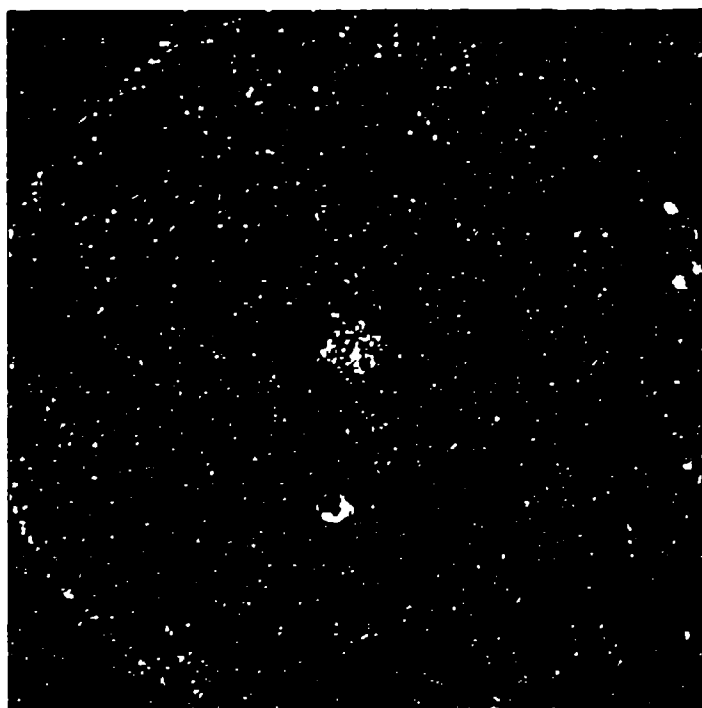
Figure 2.44 -- 4867Å filter, Int: 3, CCD: HI: **Exp: 1 sec ƒ11 5R-sec**
[Stretched: Black = 23, White = 45]

Figure 2.44 demonstrates the increased coupling for single photon-events by increasing the intensifier gain by a factor of 14. From Section 2.10.1, we would expect each

detected 5577Å photon to give a signal of around 55 to 220 ADUs. Comparing this image to **Figure 2.43**, we can see that the increased coupling does increase the number of pixels filled in the source area of the image.
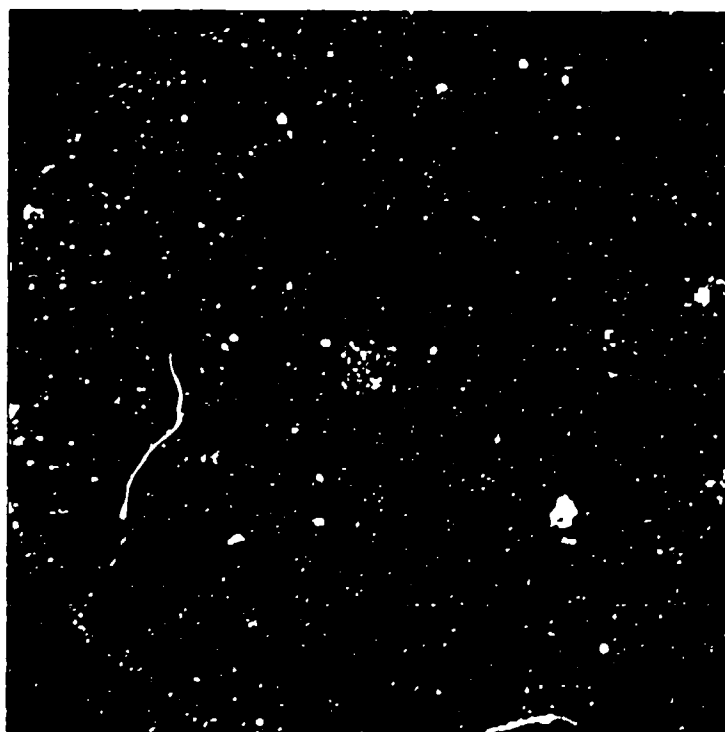
Figure 2.45 -- 4867Å filter, Int: 3, CCD: HI: **Exp: 1 sec ƒ16 2R-sec**
[Stretched: Black = 23,   White = 45]

Figure 2.45 is a 2R-sec image taken with a 1 second exposure. Again, the results of this image agree with the analysis in Section 2.10.2 that roughly 10% of the pixels should be filled with photon-events

A 5R-sec image was taken with the intensifier gain on minimum and the CCD gain at LO in which the source was questionably detectable. As discussed in Section 2.10.1, the photon-event coupling is weak in this gain setup, and thus decreases the *perceived* sensitivity of the instrument. The fact that a 5R-sec was rot really detectable at LO Gain, but easily seen at HI Gain confirms the decreased coupling. For situations where there are very low photon counts, the CCD should be used at HI gain.

Similiar images were t·ken for the other filters on the HAARP imager to check sensitivity across the spectrum.

70

2.10.4 Sensitivity for the HAARP Imager

Quoting sensitivities is always a little bit of a ·cky issue. One can define sensitivity in many ways. Typically, KEO has tried to deter.. ne the minimum source illumination that is required to visuall·/ detect the source object in the image.

Even this signal to noise is fairly subjective. For this low intensifier gain image, statistics of a box inside the source area yields an average pixel value of 22.84. With the light source off, the average pixel value is 20.79 with a Bias average of 19.8. Looking at the statistics of the same source with the intensifier at maximum, we get an average pixel intensity of 50.63, with the background averaging 22.50. This says we have a much better signal to noise, but actually it only looks better because each photon creates a much higher pixel event at maximum gain.

From the above analysis, it appears that the most significant measure of sensitivity is probably what percentage of pixels get filled in the image. From the analysis in Section 2.10.2, we see that a 5R source at 4867Å, should fill about 20% of the pixels. **Figure 2.44** confirms this result and points out that it is fairly easy to visually detect a source in this situation. However, even at 2R (or 9% of pixels filled) in **Figure 2.45** we can see that the source is detectable, hence the subjectivity.

The minimum sensitivity should be a function of the photocathode response, and it was measured for the HAARP imager at the following wavelengths and estimated at:

Wavelength	~10% events	~20% events
4282	4 R-sec	8 R-sec
4867	2.5 R-sec	5 R-sec
5300	2.5 R-sec	5 R-sec
5581	2.5 R-sec	5 R-sec
6304	4 R-sec	9 R-sec
7778	8 R-sec	16 R-sec

Chapter 3 Hardware

3.1 Overview

The HAARP Imager viewed as a black box is a very basic instrument. There is an AC power input, and just one cable that connects the instrument to a computer. This cable consists of two components - communications and data. The communications port is an RS-422 port set up for 9600 Baud, 8 Bits, 1 Stop Bit, and No Parity. Thus, to any connecting device, the camera looks like a terminal running the FORTH environment and can be treated as such. The data component of the cable has all the data lines and control lines necessary to transfer data to a 12 bit digital input frame grabber board. In our case this frame grabber is Imaging Technology's AFG board. The data signals are:

D0+/D0- through D11+/D11-	; Differential data lines
Pixel Clock+/Pixel Clock-	; Pixel Clock
Line Enable+/Line Enable-	; Line Enable
Frame Enable+/Frame Enable-	; Frame Enable

The HAARP Imager can be thought of as a standard terminal, and a somewhat generalized image data port. In theory, the imager can be controlled by any computer and a wide array of image processing boards. There are presently about 4 or 5 different compatible boards for the IBM PC architecture, as well as boards available for the MacIntosh and VME based systems.

3.2 CCD Camera Control System

The CCD camera head was custom designed and built for KEO by Advanced Technologies (Tuscon, AZ). The HAARP control electronics and software were developed by KEO to meet the specifications of our HAARP design proposal and integrated with the Advanced Technologies system. The core of the Advanced Technology design is a 4-board/motherboard located inside the imager (see Block Diagram, **Figure 3.1**). The HAARP hardware system is similiar to the MIP camera developed by KEO Consultants for Phillip's Lab Ionospheric Applications Branch (PL/GPIA).
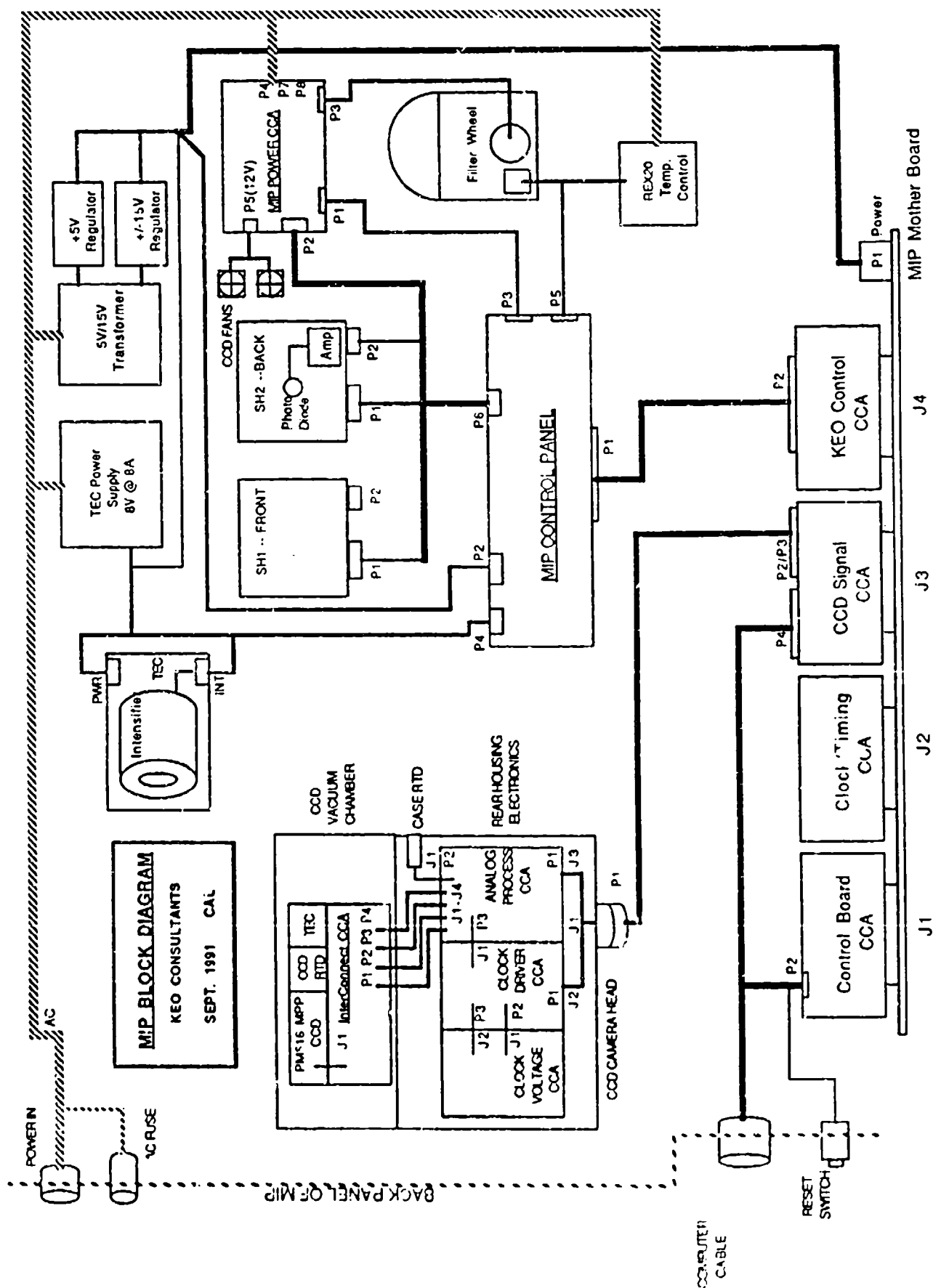
Figure 3.1 HAARP Imager Hardware Block Diagram

73

The control board consists of the system's processing units. A 68HC11 embedded processor is used as the control computer for the system (**Figure 3.2**). The 68HC11 handles all the communication to the outside world and all control units inside the HAARP camera. The 68HC11 talks directly to the host computer via a serial interface. It monitors temperatures and other analog parameters via its 8 channel 8-bit ADC. Using it's multiplexed parallel inputs, the 68HC11, talks to the DSP 56001 on the Advanced Technologies control board, and in addition talks to the KEO Interface control board which controls the imager's filter wheel, intensifier and shutters. The DSP chip controls
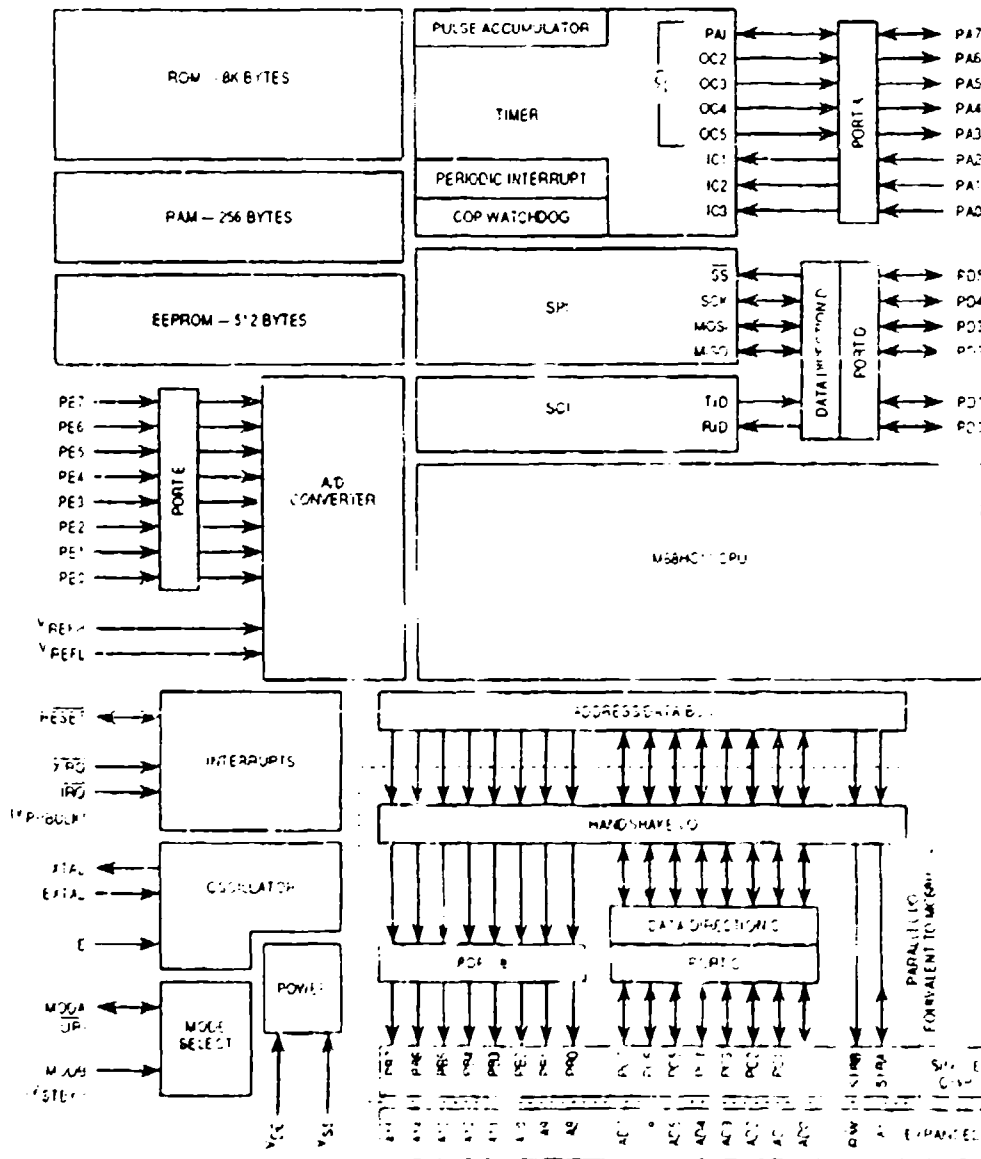
Figure 3.2 68HC11 Architecture

74

all timing states and control signals for the CCD camera head. These signals are used on the Clock/Timing board and then sent to the Signal board where they are sent out to the CCD camera head. The Signal board also takes the imaging data and control lines described above, and sends them out to the host frame grabber.

3.3 CCD Camera Head

The Advanced Technologies CCD camera head consists of two sections. The back part of the unit contains all the analog and interface electronics for the CCD. The front part of the unit is an evacuated chamber containing the CCD chip and the thermoelectric cooling head (TEC). The warm side of the TEC is coupled to the body of the unit and is cooled both convectively (the back section) and with forced air (two small fans in the front section). The TEC can normally produce a temperature differential of about 65°C.

The back part of the CCD camera head should not be opened or serviced except by qualified personnel. The front section should NEVER be opened. The CCD is extremely sensitive to static and is a very expensive device. The vacuum in the front section is extremely important for the operation of the head both to eliminate condensation on the CCD chip and for temperature insulation. If this vacuum is lost and air is let in, the CCD will frost up with ice and become unusable. **Opening the front section will void all Advanced Technology warranties.**

The CCD head electronics are critically set for optimum noise reduction. There are many critical voltages set that are tailored to our particular CCD and any tampering with these voltages will result in degradation in performance of the CCD. The electronics should NEVER be disconnected from the CCD Interconnect Circuit Card Assembly (CCA) except by the most skilled engineer and under very controlled environmental conditions.

There are many test points on these electronics cards with which you can check the operation of the camera head. It is suggested, however, that if there is a problem, the CCD head be returned to Advanced Technologies for servicing.

3.4 The CCD Chip

The HAARP Imager uses a Photometrics PM516A CCD. This chip has a 516x516 array of 20μm pixels of which 512x512 are captured by the frame grabber. The well depth of the pixels is about 215,000 electrons. The PM516 is set up to operate in MPP mode which is a CCD configuration that allows a 30 times reduction in dark noise. This is attained by using a boron implant that creates an intrinsic voltage in the wells to reduce dark noise buildup. Specific performance of the CCD chip can be found in Section 2.1.

3.5 KEO Control System

The KEO control system uses the main processor, the 68HC11 of the Advanced Technologies system, to control all operations of the HAARP camera. The control components are:

Shutter 1 (front)
Shutter 2 (back)
Image Intensifier
Filter Wheel
Temperature Monitoring
Intensifier Average Brightness   (detector located in Shutter 2)

All these components are controlled via software in the FORTH environment and executed by commands from the host computer. The 68HC11 then passes on the appropriate command signals to the KEO Interface control card, which is the fourth card on the motherboard.

3.6 KEO Control Interface Card

The Interface card is arranged in two sections: analog and digital control. The analog section consists of five analog amplifiers monitoring the following:

CCD temperature   Filter Wheel temperature
TEC temperature   Intensifier temperature
Intensifier brightness

These voltages are scaled to 0-5V and are always present to the analog digital converter (ADC) inputs of the 68HC11 (PortE). A detailed description of the temperature amplifiers will follow. The analog section also supplies a 2 mA constant current source (OP37) for the four RTD sensors. The sensors are connected in series and hence **any break in the circuit will result in none of the RTD's working.** There are three jumpers on both the Interface board and the control panel CCA to bypass selected RTD's and isolate different circuits. The analog section also creates two reference voltages to scale the temperature voltages.

The digital section of the board consists mainly of controls for the filter wheel and the image intensifier. There is an address decoding generic array logic device, GAL (U15), which interfaces the board to the 68HC11. The image intensifier circuit selects from either computer control or manual control one of four 50K variable resistors (VR5-VR7) used by the image intensifier to set the HV gain of the tube.

The filter wheel control consists of an embedded stepper-motor controller chip (U8) which contains an EEPROM (U17) with the control program. There are interface electronics which select 3 bit commands either from the 68HC11 or manually from the control panel to select the filter wheel position. An EPROM (U7) constantly monitors the state of both this command input and the filter wheel position and when there is a difference issues the appropriate movement command to the CY545 controller (U8) which then steps the filter wheel into the new position.

Presently, the EEPROM program just stops if there is a problem with the position or the command location. The logic behind this is that if there is a problem in either of the systems, we do not want the controller to endlessly try and position the stepper motor and perhaps result in a catastrophic failure. [We could add a control line from the 68HC11 that can position the filter wheel. This would give an independent, host-controlled system to control the filter wheel in the event of some failure in the interface electronics.] There is a reset switch for the CY545 Stepper Motor controller located on the Interface Board.

The Interface Board is delivered with the following jumper settings:

J1 -- OUT   ; Control Panel RTD Excitation Current Bypass

J2 -- IN   ; RTD Excitation Current Source

J3 -- OUT   ; CCD Head RTD Excitation Current Bypass

J4 -- IN   ; Analog and Digital Ground Common Point

3.7 Control Panel

The control panel, which is accessible from the outside via a magnetically closed panel cover, contains the filter wheel temperature controller, and all the electronics to manually control the shutters, the image intensifier, the filter wheel. All the power supplies and system temperatures can be monitored on the control panel. In addition, there is an opto-isolated, voltage shifting interface which sends the shutter and stepper motor control signals to the KEO Power I/O board. This interface isolates any power supply noise of the shutters and stepper motors from the analog voltages which are used in the CCD camera head (and are very noise sensitive).

3.8 Power I/O board

The Power I/O board has the shutter drivers, the stepper motor drivers, and a 12V supply for the CCD cooling fans. Both the HAARP and the MIP Imagers have been upgraded to use the Power I/O Rev.A Board. There are five fuses on the circuit board to protect the input transformer (AC), each of the shutter coils (32V peak, 4V holding) and stepper motor coils (24V unregulated). During parked modes for the filter wheel, the coils are turned off via the Stop interface signal and the intrinsic friction of the motor and gears is used to hold the FW in place. This saves on heat/power dissipation of both the driver (UCN5804B) and the stepper motor coils (each 24V @ 1A or 48W).

The RevA version of the Power I/O board has diodes between the UCN5804B and the StepperMotor. This was necessitated by a change in the manufactured configuration of the UCN5804B. In addition, dropping resistors were inserted between the shutter and the shutter supply to allow for different shutter coil resistances (i.e. switching from UniBlitz shutters to Melles-Griot).

3.9 Filter Wheel

The HAARP filter wheel has three basic components: stepper motor, position monitoring, and temperature control. The stepper motor connects directly to the Power I/O board. The HAARP Imager has been set up using the stepper motor on the filterwheel in half-step mode, which increases resolution and results in much smoother operation. This modification directly pulled pin 10 of the UCN5804B on the PowerI/O board up to Vcc (or pin 16).

The position monitoring electronics involves four transmission switches. These are connected to the control panel and then to the KEO Interface Control Card. There are three position switches which give a 3 bit position code: 1-5. The fourth transmission switch gives a pulse for every filter position and is used to clock the position code into D-flip-flops and subsequently into the command EPROM. In addition, there is an RTD used to monitor the FW temperature. This signal returns to the Control Panel CCA where an instrumentation amplifier (U13) amplifies it and sends it to the KEO Interface Control CCA.

On the front of the filter wheel is a connector that has a mating photoconductive cell. This is used to inhibit the image intensifier power supply from switching on in the event of too much ambient light. The sensitivity of this light detector can be set by adjusting the 1M pot on the KEO Control Panel CCA (VR1)

The temperature control consists of a REX C100 temperature control unit and is independent of the KEO interface electronics. An RTD is connected in the three wire configuration, and the outputs of the REX C100 connect to an SSR inside the Imager chassis and subsequently to an AC heater pad glued to the inside of the filter wheel. For operation of the REX C100, see the REX C100 user's manual.

3.10 Image Intensifier Control

The image intensifier is housed inside a thermo-electrically cooled housing that cools only the input cathode to about $25^\circ C$ below ambient, thus reducing image intensifier dark current by a factor of about 10-15. There are gas connection fittings on the side of the image intensifier cooler to flush the input window and image intensifier fiber optics input with dry N_2 in case of condensation in humid environments.

Note: **Due to a fabrication error by Products for Research, the connectors to the HAARP Intensifier Cooler are different than the MIP Intensifier Cooler. Because of this, the two coolers are NOT directly interchangable.**

Power to the TEC comes from a Photometrics unregulated TEC supply. RTD connections are routed into the power cable on the intensifier housing labelled PWR and connect to the KEO Control Panel CCA where an instrumentation amplifier (U14) amplifies its signal and sends it to the KEO Interface Control CCA. The image intensifier is a 25mm GenII VARO tube and connects to the KEO Control Panel CCA via the housing connector labelled INT.

The intensifier cable has the standard 4 wires to connect to an image intensifier - Power, Gnd, and two gain connections. The intensifier has 4 gain settings, 0-3, which can be controlled either by the computer or via the manual controls at the HAARP Imager. The power is supplied by a 3V supply on the KEO Control Panel board and the ground returns through a 5 Ohm resistor which monitors the actual current through the intensifier and activates an LED on the control panel. (An improvement would be to connect this to a TTL status signal to the host via the 68HC11 interface.) The gain signals return to the KEO Interface Control CCA and goes through the currently selected 50K potentiometer.

3.11 Shutters

There are two shutters in the HAARP system. Shutter 1 is defined as the front shutter, closest to the filter wheel, and blocks light to the front of the image intensifier. Shutter 2 is defined as the shutter closest to the CCD head and blocks light to the CCD. The shutters can be controlled either by the computer or manually. The shutters are each fused with a 1A SB fuse to protect the shutter coil. These fuses are located on the Power I/O board in the Imager.

The shutters used in the HAARP Imager are Melles-Griot #04-IES005 shutters, and are at a location in the optical path so as to minimize exposure spatial non-uniformity resulting from the finite shutter speed. Each shutter has two connectors: one for the solenoid and micro-switch, and one for a photo-diode amplifier board. Only the back shutter (Shutter2) has a photodiode amplifier connected. This is used to measure the luminance of the intensifier phosphor.

The Melles-Griot shutter control uses a 6V Relay to open and close its blades. The shutter is specified at ~4x opening voltage and 1/2x holding voltage. The KEO PowerI/O board creates a 32V (unregulated) 80 msec pulse using a timer (U1) and the UCN2944 Supply side driver (U2). The holding current is supplied by the LM317 voltage regulator and is set for around 5.5V, giving a holding current of about 3.5V after the Darlington and diode voltage drops in the electronics.

On the Power I/O board Rev.A, there are also dropping resistors between the supply and the shutters giving an additional voltage drop to the shutter. The HAARP Imager has 3.0Ω dropping resistors which bring the final starting pulse voltage from 24V to 16V and the holding voltage from 3.5V to 2.4V. (This drop from the original specs of the Power I/O board is needed because we changed from the UniBlitz shutter which had a 12.5Ω coil to the Melles-Griot which uses a 7Ω coil.)

A micro-switch is mounted inside the shutter so that it closes when the shutter blades are fully open. This shorts to ground a circuit on the KEO Control Panel CCA which puts out a TTL signal and also controls an LED display on the control panel. Thus there is visual feedback as to whether the shutter is open, and a TTL signal which is returned to the 68HC11 processor and can be read by the Host for a shutter status indication.

The photodiode output is amplified by an AD515 amplifier which must be located very near the photodiode because of the low currents being measured (pA). The photodiode was selected for its relatively fast response and fast dark decay time. The output of the AD515 is returned to the KEO Interface Control CCA via the KEO Control Panel CCA and is amplified with variable gain (VR4 on U9) and sent to the ADC port of the 68HC11 (PE4). This variable resistor is set so that 5V corresponds to the image intensifier operating in AGC mode. Reading this photodiode gives the Host a measure of the average luminance of the image intensifier phosphor and from this analog information it can then be decided what exposure and/or gain is most appropriate for the next image acquisition. It also allows the Host confirm that the intensifier is not operating in the AGC mode where intensifier output is no longer linear with light input.

Keeping Shutter 1 closed during the exposure allows the Host to obtain a dark image that includes both the dark noise of the CCD and the image intensifier. During a normal exposure, the front shutter is opened first and checked. The Intensifier brightness

is then checked to make sure that it is operating in its linear region. Finally, the back shutter is opened, checked and the exposure interval begins. At the end of the exposure interval, both shutters close simultaneously and are checked.

3.12 Power Supplies

AC power for the HAARP Imager is supplied to four separate transformers:

    (1)   REX C100 power supply

    (2)   5V, +/-15V power supply transformer

    (3)   TEC Unregulated power supply

    (4)   KEO PowerI/O CCA power supply

(1) The REX C100 is fused with a 1A SB fuse. It's power consumption is only 15VA but AC to the REX C100 also supplies the Filter Wheel heaters.

(2) The 5V, +/-15V power supply transformer is taken from a PowerOne power supply and is rated at 5V @3A, +/-15V @1A. The transformer is fused with a 1A SB fuse. The 5V regulator is mounted vertically inside the HAARP chassis and the +/-15V regulator board is mounted in the box on top of the chassis with the regulators heat sinked directly into the HAARP chassis from the inside.

(3) The TEC's in the HAARP are powered by a Photometrics unregulated power supply and fused at 1A SB. To handle both TEC's, the input coils of the transformer are connected in parallel to increase the current capatity of the power supply. The image intensifier TEC and the CCD TEC together draw about 8A at around 7.5VDC. They are connected in parallel to the output of the power supply.

(4) The KEO Power I/O CCA has a 20VAC 56VA transformer and is fused on the circuit board with a 1A SB fuse. When drawing no current, the DC unregulated power supply charges up to about 32-35VDC. This is used as a starting pulse for the shutters. The stepper motor draws 1A per phase. At 2A, the unregulated power supply puts out 24VDC (which is the voltage specified for the motor). There are also two LM317 voltage regulators. U4 is set to 5.75V and is used for the CMOS logic supply and shutter holding voltage. The shutter coils are 7 Ohms, so at 2.4V they each draw 340mA. R10, R11A & R11B are used reduce the input voltage from 32VDC so that the LM317

82

doesn't dissipate as much heat. U5 is set to 12V and is used to power two DC fans (.1A each) connected in parallel, and used to cool the CCD head.

3.13 Back Panel

The back panel has two connectors: AC and Computer Interface. The AC plugs into any 110-120VAC outlet. The Computer Interface connecter contains all the signals for the RS-422 communications, as well as the 12 bit digital image and clocking signals. All signals on this cable are differential which allows longer transmission distances and noise immunity. The main power fuse for the HAARP Imager (2A SB) is located on the back panel. A power on/off switch is also located on the back panel.

A recessed red button is located by the Interface connector. This is the reset button for the 68HC11 processor inside the HAARP. Pushing this will reset all the electronics and bring the instrument up in it's initial state. Some of the electronics that are independent to the processor will not be reset by this button, and you may need to actually turn off/on to achieve a full reset.

3.14 Temperature Monitoring Equations

The KEO Interface Control CCA provides a 2mA excitation current for the RTDs. The 8 bit ADC (0 to 5V) gives 256 steps. If the resolution is 0.5°C, then the range is 128°C.

Temperature Range: -50°C to 78°C
DIN RTD curves: R(-50°C)=80.3Ω   R(78°C) =130.1Ω

0 to 5 Volts/128C   =   38 mV/°C.

With 2mA excitation: V(-50°C) = .161 Volts, V(78°C) = .26 Volts
                             V(78°C) - V(-50°C) = 99 mV.

Set overall system gain to 50.   LF347 Gain = 2,   1N101A Gain = 25
Gain of 1N101A   = 1 + 40K/Rg: Rg = 1.67/KΩ

Vref = .16 Volts x 25 = 4.02 Volts Offset for LF347 to scale -4.02 V to 0 Volts

To display the temperature on the HAARP Control Panel, an ACCULEX DP350 DVM is used. The inputs to the 68HC11 ADC are scaled 10mV/°C to read degrees centigrade. So -50°C = 500mV or 0.5V. The decimal point of the meter is shifted right two places so the meter actually displays degrees Celsius.

0 - 5 VDC = -.5V to +.78V   Gain = .256
An LF347 (U6) on the Control Panel CCA with R2/R1 =5.11KΩ/20KΩ = .255
An offset voltage is needed to scale 0V down to -.5Volts:
.5/.2555 = 1.96V   LF347 (U9) on Interface CCA using VR1 to set offset.

Chapter 4   Image Processing System

The HAARP Imager runs on an IBM AT system under the Windows 3.1 Operating System. The casual user should not have to know any details of how the camera works, so this section will mainly focus on the computer and image processing systems.

Since the HAARP software runs under Windows 3.1, it is fairly user-friendly. The operator should be able to become proficient in the basic operations without a detailed knowledge of the system software. However, it is suggested that to obtain maximum use out of this powerful system, the interested user should become familiar with the following systems:

Computer System	Suggested Reading
DOS 5.0	*MS-DOS Reference Manual*
Windows 3.1	*Windows 3.1 Reference Manual*
ITI Adv. Frame Grabber	*ITI AFG Software/Hardware manual*
	ITEX V2.2-2 Release Notes

Camera System	Suggested Reading
HAARP Camera	*HAARP Hardware Reference Manual*
MaxFORTH	*MaxFORTH User Manual*
68HC11	*Motorola 68HC11 User Manual*
DSP56001	*DSP56001 User's Manual*

4.1 Overview:

The HAARP Imager can be thought of as composed of two separate components: the camera, and the image processing system. The camera is controlled by the image processing system through a control program called:

MIPCTL

It is also possible to control the camera with any communications program if the communications protocol is set as follows:

85

9600 Baud 8 Bits 1 Stop Bit No Parity

The camera has an RS-422 serial port which is connected to a serial board in the computer to COM4. The printer port is set up for COM2. Both these ports are labelled on the back of the computer.

The image processing system is centered around hardware called the Advanced Frame Grabber (AFG) made by Imaging Technology. This is a two board set with a daughter board that resides inside the IBM AT computer. This board is controlled through the program MIPCTL, and it can also be controlled manually through the ITEX Interpreter (INTRP) supplied with the board. The camera can thus be controlled by using both a communications program and INTRP in tandem. This can be useful in developing new algorithms, or in being able to take data in a way that the MIPCTL does not support.

The control program MIPCTL sets up the communications with the camera and enables the user to control all the basic features of the camera. In addition, this program gives a set of commands for the AFG board and data acquisition and analysis functions. The MIPCTL program is viewed as an evolutionary program, and it is expected that it will be developed further to add extended functionality as dictated by experience and field use situations.

4.2 IBM AT Image Processing Hardware:

The HAARP Image Processing System is based upon an IBM 486 processor. A Gateways2000 486DX2/50MHz machine is used, but any AT system with a 386 or greater would be satisfactory. The following is a brief description of the AT's hardware configuration.

Floppy Disks:

Both a 3.5" and 5.25" floppy disk are mounted in the system. The HAARP IP System uses a dual 1/2 space 3.5"/5.25" TEAC floppy drive FD-505 supplied by JDR Microdevices configured for the default operation A: 1.2MB and B: 1.4MB.

650MB Read/Write Optical Disk:

A RICOH Magneto-Optical Drive RO-5031E (SN: K9101148) with a SCSI interface is mounted internally. This is fully compatible with the ASIPII Optical Disk system and the MIP Optical Disk System on other Air Force GL imagers which both use the SONY M/O drive. Because of the increased rotational speed of the RICOH drive, it is recommended by RICOH that only SONY media be used with this drive (EDM-1DAOs: 1024Bytes/sector, 650MB optical disks). The RICOH drive is unterminated as there is also a hard-drive connected to the SCSI port in the HAARP IP System.

An APT Odessa SCSI interface board (SC1000) is used to interface the optical disk to the AT. This is fully DOS compatible and can be used as any other drive in the system. The SC1000 is installed using IRQ10 and memory address segment D600:0000. The Odessa system comes with drivers MOD.SYS and MOD.EXE. The HAARP Imager has v1.26 installed. The switch option /4 should be used when running MOD.EXE which utilizes the smaller cluster sizes (8192) of DOS4.01 and greater. M.BAT assures that this switch is used.

A 650MB MO disk can hold about 2400 2x2 binned images per side (1/8MB per image) or about 600 1x1 binned images per side (1/2MB per image). On the HAARP computer system, the MO drive is set up to start partitions at drive E:

766MB SCSI Hard Drive:

An additional hard drive was installed into the HAARP imager to allow for rapid image processing of large data sets. The 766MB hard drive has enough room to store two sides of an M/O disk (or one side of raw data and it's associated transformed images). This disk was also installed to assure backup in the event of the IDE Hard Drive or the RICOH MO drive failing. Another use for this drive is to collect large amounts of data quickly when the write-time of the RICOH MO drive is too long to take 'real-time' data. After the data has been taken on the SCSI hard drive, it can be backed up onto the MO disk when time is not critical.

A Seagate WREN 6 SCSI drive (ST4766N) was chosen for the HAARP imager and installed internally in series with the RICOH MO disk. This disk is terminated as it is the last drive in the SCSI chain. The ST4766N is a full-height disk drive.

This drive can hold about 5800 2x2 binned images (1/8MB per image) or about 1450 1x1 binned images (1/2MB per image).

200 MB IDE Hard Drive and I/O Ports:

A 200 MB Hard Drive is mounted internally in the system with an IDE controller (DFI IDEIO). This controller also has provisions for the parallel interface brought out to the back of the computer and two serial interfaces set up as COM1 and COM2. COM1 is currently set up for a serial Microsoft Mouse, and COM2 is accessible for peripherals such as a serial printer.

RS-422 Serial Interface Board:

A B&B Electronics 3PXCC1A 1PORT RS-422 Board is used to interface to the HAARP Imager. This board was chosen because it allows the selection of any available I/O address for the COM address rather than the standard COM port addresses. In addition, it allows selection of IRQ's up to IRQ15. This flexibility is needed in the HAARP IP System because the VGA board supplied in the Gateways 486 (ATI Ultra) uses the COM4 I/O address for it's own purposes.

The 3PXCC1A was installed using COM4 @ IRQ11, and the I/O address 0x310. The COM4 settings must reflect this under Windows 3.1 in order for this to work.

Gateways Telepath Modem/Fax Board:

A Gateways Telepath 3.5 Modem/FAX board was installed in the HAARP IP System and allows up to 14.4KBaud Modem communications and 9600Baud Send/Receive FAX capabilities. The Telepath is installed at COM3, IRQ5 and has both communications and FAX software installed with it. For Windows 3.1 users, CrossTalk for Windows (communications) and WinFax (Faxing) can be used.

SuperVGA Board:

The HAARP has an ATI Ultra Board with 1MB of Video RAM installed, which has a resolution of up to 1024x764. The output of the VGA board is sent to the video mixing input connector on the AFG board.

Advanced Frame Grabber (AFG) Image Processing board:

This two board set contains all the image processing and display hardware for the HAARP Imager. The model number for the AFG board ordered for HAARP is: VP1510-KIT-512-U-C1-M3-AT. A GSP chip (TMS34010) controls all the display and system interface functions on the board. This GSP has 2MB of system memory of which about .5M is currently used. 2MB of video frame memory (1Kx1Kx16) make up the frame buffer. Four multiplexed RS-170 inputs and one 12 bit digital input port are available on the board. The 12 bit digital input port is used for the HAARP Camera head. The AFG also has an extensive hardware Arithmetic Logic Unit. It is recommended that the user read the AFG Hardware and Software Introductions to become familiar with the board's capability. A block diagram of the AFG board is shown in **Figure 4.1**.
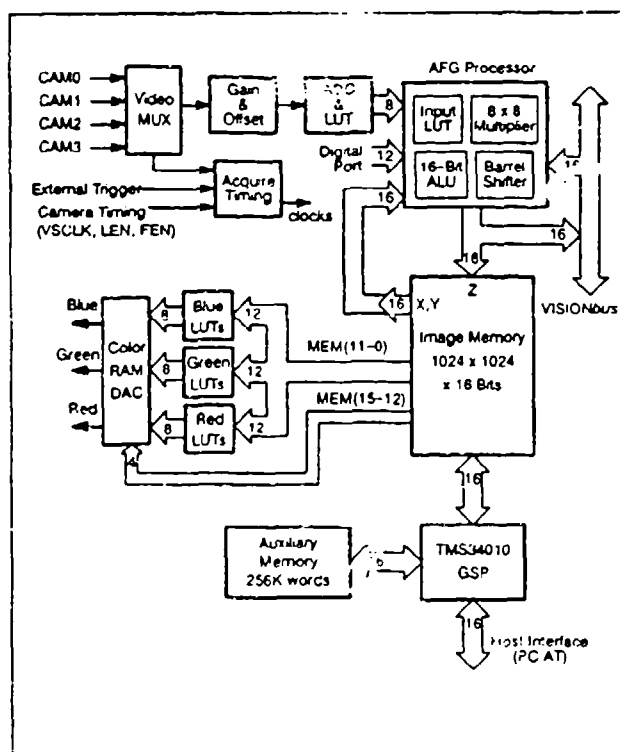
Figure 4.1 -- AFG Block Diagram

89

The AFG has a daughter board that mixes the VGA from the VGA board with the output display from the AFG's frame buffer. This output must go to a multi-sync monitor such as the NEC 3FGx. Using this VGA mixing board, one monitor can be used to view both the image display and the computer's screen. A key color is set up to choose how to mix both displays. Two displays can also be used at any time for greater flexibility.

NEC 3FGx Monitor:

The NEC 3FGx is a 15" Multi-sync Flat-Screen display capable of up to 1024x768 resolution. It is presently set up for VGA display (640x480). The NEC 3FGx monitor delivered with the HAARP Imager is Model #: JC1532VMA-N and SN: 22M24855A.

Gateways 2000 486 Computer:

The computer runs an Intel 486DX2 processor at 50MHz with 8MB of RAM. DOS 5.0 and Windows 3.1 are installed on the system. DOS 6.0 is provided with the system if an upgrade is desired. It is recommended that the operator become familiar with these operating systems (especially Windows).

4.3 Understanding the Image Processing System

4.3.1 AFG Hardware Specifics

Hardware of the Image Processing System is mostly defined by the frame grabber board: the Advanced Frame Grabber from Imaging Technology (henceforth referred to as the AFG). It is extremely beneficial to have a basic understanding of this board and the ITEX documentation has a good introduction into both the hardware and the software for the board, especially Chapter 3 of the Hardware Manual.

The AFG Board has a 1024x1024 pixel frame buffer with 16 bits depth. The lower 12 bits are used to store the image information, and the upper 4 bits are used for graphic overlays such as image labelling, lines, graphs, etc. In this way, the image labelling does not affect the integrity of the data. There are times however, when one would like to use all 16 bits of the frame buffer for image data, such as when adding ten 12-bit images

together to take an average. Care then has to be taken when using the frame buffer to know exactly what section you are using and what is presently available.

The version of the MIPCTL software delivered with the HAARP IP System has provision for doing all image overlay and graphics in a super-imposed VGA window so that none of this data is being written into the frame buffer memory. This insures the integrity of all 16 bits of image memory.

4.3.2 Image Definitions: General Area of Interest (GAOI)

An area defined in this frame buffer is called a "General Area of Interest" or abbreviated as GAOI. There are defined GAOI's that are used in the MIPCTL application that are useful to know. The frame buffer is broken into four quadrants:

UL:	Upper-Left
UR:	Upper-Right
LL:	Lower-Left
LR:	Lower-Right

There are four predefined depths used with the MIP application:

8 Bits
12 Bits
16 Bits
Overlay Plane (B12-B15)

Predefined abbreviations are used for these area and use the nomenclature:

GUL12 = "Generalized Area of Interest, Upper Left quadrant,
12 bits deep" would have the coordinates (0,0) to (512,512)
GLROVL = "Generalized Area of Interest, Lower Right quadrant
Overlay" would have the coordinates (512,512) to (1024, 1024)

This nomenclature is used throughout the MIPCTL application.

91

4.3.3 Image Memory definitions for MIPCTL applications

Since the casual user in the field does not need to be concerned about the above nomenclature, a number of image areas have been predefined for the MIPCTL application, which are optimized for the kinds of images that the instrument is likely to be taking. Understanding this predefined structure is central to using the MIPCTL application.

Since the HAARP and MIP imagers have a 512x512 CCD digitized to 12 bits, we can define two types of likely images to be collected by the instrument. At full resolution, we have 512x512 pixels, and at 'low-resolution' we use a 2x2 binning which combines 4 adjacent pixels into one 'super-pixel'. This achieves a lower resolution of 256x256 pixels, and quadruples the dynamic range of the CCD (not the dynamic range of the ADC converter which being 12 bits is always 4096:1).

For high resolution (Hi-Res) images, we define four images that can be stored into the frame buffer memory. As in Section 4.3.2, we define these four images as UL, UR, LL, & LR ("Upper-Left, Upper-Right, Lower-Left, and Lower-Right"). Each of these images are 512x512 pixels. There are four images in the Hi-Res **Quadrant** which is 1024x1024 pixels wide. This happens to correspond to the whole frame buffer. Thus, when using four Hi-Res images, there is no room for any more images on the AFG board.

For low resolution (Lo-Res) Images, we define the same four images per **Quadrant** each with the dimensions 256x256 pixels. Because each quadrant now has a dimension of 512x512 pixels (note: the same as a Hi-Res **Image**), there can be four Lo-Res quadrants defined in the AFG frame buffer. These four **Quadrants** are defined as:

Data:	"Upper-Left" usually used to collect raw data images
Back:	"Upper-Right" usually used to collect background images
Scratch:	"Lower-Left" usually used for calculation images
Display:	"Lower-Right" usually used for display images

In each of these quadrants, there are the four Lo-Res images defined above.

Within each of these images, there are four depths that can be defined:

8 bits   **12 Bits**   **16 Bits**   **OVL** (4 Bits: B12 -- B15)

These depths are self-expanatory. The most common depths used in MIPCTL are **12** bits for operations on 12-bit images (e.g. raw-data), and **16** bits for operations on transformed data (e.g. stretched, added, etc.).

The nomenclature developed for MIPCTL to identify these different images is as follows:

QQ:IIDD   where:

QQ = Quadrant:	HR == Hi-Res	
	DA == Data	(Lo-Res, Upper-Left)
	BK == Back	(Lo-Res, Upper-Right)
	SC == Scratch	(Lo-Res, Lower-Left)
	DS == Display	(Lo-Res, Lower-Right)
II = Image	UL == Upper-Left	
	UR == Upper-Right	
	LL == Lower-Left	
	LR == Lower-Right	
DD = Depth	8 Bits	(B0 -- B7)
	12 Bits	(B0 -- B11)
	16 Bits	(B0 -- B15)
	OVL: 4Bits	(B12 -- B15)

Thus, using this nomenclature, the image:

SC:LR12   corresponds to:

"Scratch Quadrant: Lower-Right image, 12 Bits deep"

Using pixels locations in the frame buffer, this would be an image:

Origin: X = 256, Y = 768   DX: = 256   DY: = 256

This image is represented by the following diagram:

SIZE 12

This diagram represents the MIPCTL control box that facilitates selecting an image and is common to most of the image operations in the MIPCTL application. The four boxes represent the images: the gray box representing the currently selected image. The button at the bottom of the box has text in it that represents the Quadrant selected. Pressing this button brings up another box that allows you to select which quandrant you want:

Select Quadrant...

Quadrant
○ Hi-Res: Whole Buff 1x1
◉ Data: Upr Left 2x2
○ Back :   Upr Right 2x2
○ Scrtch: Lwr Left 2x2
○ Display: Lwr Right 2x2

Pixel Depth
○ 8 Bits
○ 12 Bits
◉ 16 Bits
○ Overlay

OK
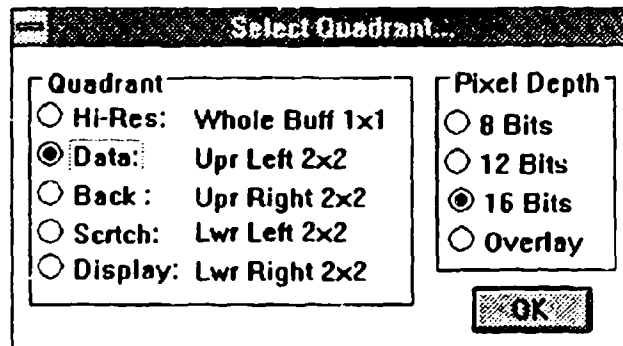
Any changes in the quadrant is reflected by the text in the button. So the above change would be reflected with the text: DATA:16.

4.3.4 Limitations in the AFG Frame Buffer

Because of the physical size of the frame buffer (1024x1024x16), there are only so many images that can fit into the frame buffer at any one time. It should be understood that physically, the Lo-Res Quadrants occupy the same physical space as the Hi-Res Images. Thus, modifying something in the image of DA:UL16 is also

modifying the data of HR:UL16. Because of this, the user has to be careful when using data of different resolutions.

The frame buffer can hold:

Four	Hi-Res **Images**		1 **Quadrant**
		OR	
Sixteen	Lo-Res **Images**		4 **Quadrants**

An example of mixing resolutions could be using the following images:

HR:UL16	HR:UR16	HR:LL16	
DS:UL16	DS:UR16	DS:LL16	DS:LR16

4.4 AFG Board Hardware Configuration and Jumper Settings
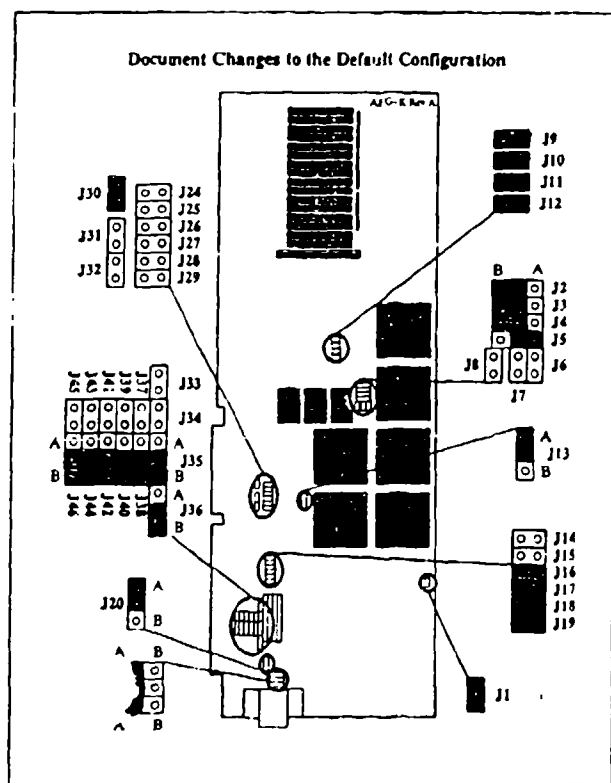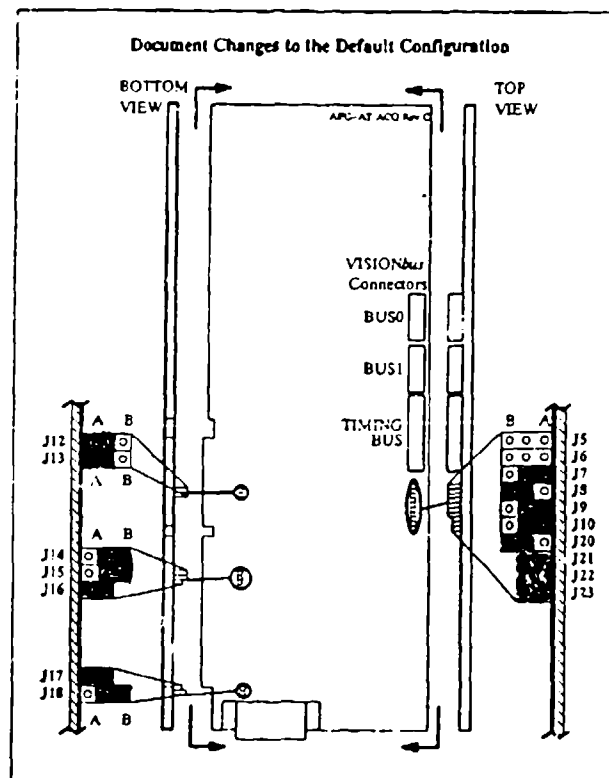
Figure 4.2 -- AFG Board Jumper Settings

95

Chapter 5 Using the MIPCTL Software

5.1 Using the MIP Image Processing System

The basic functionality of the Image Processing System will be discussed here. There is an unlimited range of possibilities that are open to users, depending on their understanding of all the components of the system. The Windows operating system, however, is designed to make applications self-explanatory to use. With a limited learning curve, the user should be able to become productive with the MIPCTL software. The MIPCTL software tries to strictly adhere to the IBM CUA guidelines so that all key strokes and mouse actions are consistent with the Windows Interface.

If users becomes limited by the constraints of the MIPCTL application, more flexibility can be gained by opening Window's Terminal program to talk directly to the HAARP imager using the FORTH commands and opening Imaging Technology's Intrp interpreter to talk directly to the AFG board using Imaging T... .nology's ITEX language. For this, the user should refer to the appropriate manuals for instruction.

5.2 Using Windows 3.1 and MIPCTL

5.2.1 Starting *Microsoft* Windows 3.1

Upon powering on the computer, it will go through it's booting procedure until finally ending up in the Windows environment. Many windows will be displayed and the Windows Program Manager will be operating. If Windows is not running, type:

win<cr>

5.2.2 Starting MIPCTL

The MIPCTL control Program is located in tl e **MIPCTL Development** group box and under it's icon the name MIPCTL should appear. To start the MIPCTL program, double click (or use the Windows key strokes) on the icon. An "hour-glass" icon will appear as the software initializes itself. There are many stages of initialization. The longest initialization occurs the first time MIPCTL is started after the computer is rebooted. This is due to the AFG board being loaded up with all of it's board-resident software. If the

computer has not just been rebooted, the initialization should only take a few seconds. Once the initialization has been completed, the "hour-glass" icon will return to the normal "arrow" icon.

The initialization of the MIPCTL software also involves setting the hardware and software parameters. A lot of these are defined in a file called

KEOCCD.INI in directory: C:\WINDOWS\

This file contains camera, communications, and software setup information based on the last execution of MIPCTL. The file allows the software to automatically come up in the same configuration as the last execution. If this file gets corrupted, the initialization may fail, giving an appropriate error message. In this case, you will need to edit KEOCCD.INI using a text editor such as Window's **NotePad**, and manually set the parameters in KEOCCD.INI to valid values. Section 5.4 will describe KEOCCD.INI in more detail.

The communications port will be set during the initialization and if successfull, a status line in the MIPCTL Window (usually shrunk so that you can't see it) will appear that indicates the status of the COMM port. The HAARP Imager was delivered using the following COMM Port Settings:

COM4, 9600 Baud, 8 Bits, 1 Stop Bit, No Parity

where COM4 was setup (using Window's **Control Panel**) to have the address 0x310, IRQ11, and the above communications settings. If there are any problems initializing the port (say perhaps, another application like **Terminal** is using **COM4** or KEOCCD.INI has been corrupted), an error message will come up. You should exit the **MIPCTL** program and remedy the situation.

Once MIPCTL has successfully opened the **COM** port, a communications status box will appear in the Window:

```
▬ ░░░░░░░░░░ MIP Camera Control ░░░░░░░░░░ ▣▣
Comm  CCD  Control  View  Analyze  Aquisition      Help

 Com: 3 R: 9600 B: 8 P: 0 S: 0
```
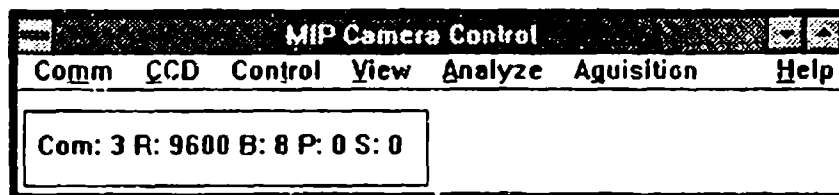
These numbers do not directly represent the communications settings, but rather the Device
Control Block Parameters that Windows uses in setting the communications port.
However, this will give you feedback as to whether the port is set up the way you want it.
Under normal circumstances, you can ignore this status window.

During the initialization, MIPCTL tries to initialize the camera to it's previous state.
If there is a problem establishing communication with the camera, the following warning
box will appear. This warning appears at any time during the application if there is a
communication problem with the camera.

```
▬ ░░░░░░░░░░ Module: SndMipCmd ░░░░░░░░░░

   (!)   Timeout Error waiting for camera


            ┌───────┐
            │  OK   │
            └───────┘
```
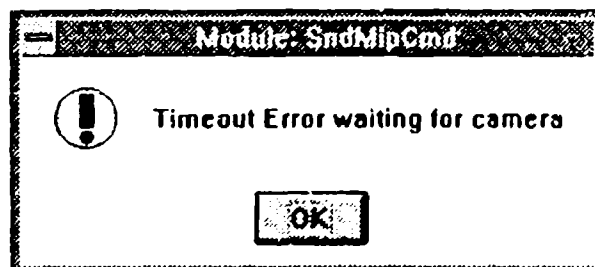
If this error occurs, check the camera communcation system: Is the cable connected to
COM4? Is it connected to the camera? Is COM4 (or whatever port you're using) setup
correctly in Windows to agree with the installed hardware? Is there another application
open that is using the COM port (such as TERMINAL)? Power down the camera (or push
the reset button on it's rear panel), and reboot the application.

5.2.3 The MIPCTL Window and *Microsoft* Windows 3.1

The MIPCTL Window is a standard Windows 3.1 window, and thus has full
Windows 3.1 functionality. You can re-size it, minimize it, move it, etc. The user should
become familiar with the use of Windows 3.1 to get maximum use out of the MIPCTL
application.

In addition, you can have any number of other applications running at the same time as **MIPCTL**, although the more you run, the slower everything will be. These applications can be any size and minimized as well. After starting **MIPCTL**, it is a good idea to minimize all other applications (including the Program Manager) and move the icons over to the right of the screen. This will give the maximum visibility for the images in the frame buffer.

5.3 MIPCTL Software Manual (v5.3.1)

If everything was successfully initialized, all the menu elements should be enabled and appear black. If they are grey that means that they are disabled. The main menu should read:

The underlined letter represents the Windows keystroke to type to activate this menu item. Type the <Alt> key, and then type the underlined letter if you do not wish to use the mouse. (see Windows 3.1 User's Manual for more information on Windows keystrokes)

5.3.1 The Comm Menu functions

The Comm menu contains the initialization functions. It looks like:
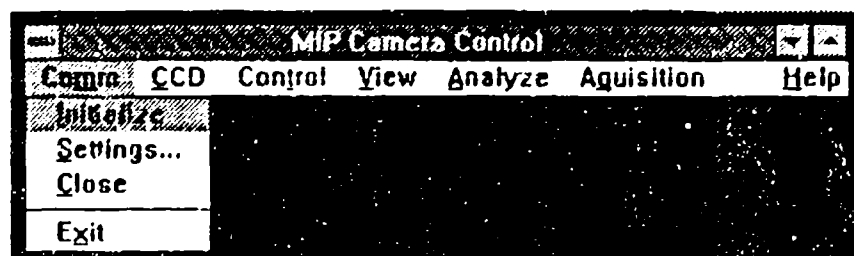
Initialize: This menu item initializes the COM port. If the COM port is already opened to the application, this menu item will be disabled and appear grey. There may be times when you would want to reinitialize your COM port. For example, you want to talk to the HAARP imager via a communications program such as TERMINAL to test some

functionality without closing **MIPCTL** and thus loosing all the information presently in the frame buffer and **MIPCTL** buffers. You would Close the COM port, and when ready re-initialize using **Initialize**. Notice that when you close the COM port, all menu items that rely on communications with the camera turn grey and are Disabled.

Settings: Settings allows you to change the communications settings on the COM port. This is normally not desired, but there could arise times when you want to use a different COM port, slower baud rate or the like. Also, this allows for future situations when the camera hardware might be slightly different. Once the settings have been changed, the port should be Closed and re-Initialized.

Settings

Baud Rate
○ 110   ○ 300   ○ 600   ○ 1200
○ 2400   ○ 4800   ⊙ 9600   ○ 19200

[OK]
[Cancel]

Data Bits
○ 5   ○ 6   ○ 7   ⊙ 8

Stop Bits
⊙ 1   ○ 1.5   ○ 2

Parity
⊙ None
○ Odd
○ Even
○ Mark
○ Space

Comm Port
○ COM1   ○ COM2
○ COM3   ⊙ COM4

Close: This closes the COM port and disables all the menu items that need communication to the camera. This is described in more detail in **Initialize.**

5.3.2 The CCD Menu functions

The CCD menu has the functions that control the CCD camera. It looks like:

```
┌──────────────────────────────────────────────────────────┐
│             MIP Camera Control                             │
├──────────────────────────────────────────────────────────┤
│ Comm │ CCD │ Control  View  Analyze  Aquisition    Help   │
│      ├──────────┐                                          │
│      │ Bias     │                                          │
│      │ Read     │                                          │
│      │ Clear    │                                          │
│      │ Default  │                                          │
│      │ Obs/Dark...│                                        │
│      │ Shade    │                                          │
│      │ Setup... │                                          │
│      └──────────┘                                          │
└──────────────────────────────────────────────────────────┘
```
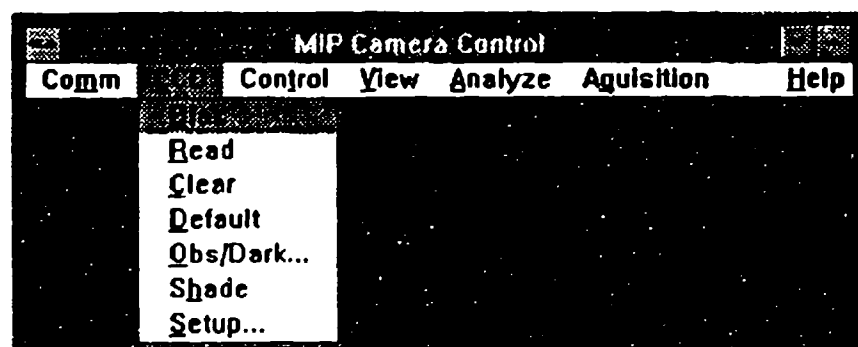
Bias: A bias is executed on the   camera and read into the frame buffer at the presently selected image (see **Display**). A Bias is used to check the read amplifier gain of the CCD read electronics. The CCD is cleared three times to make sure there is absolutely no residual charge or dark noise in the CCD wells, and the CCD is subsequently read out.

Read: The CCD is read out into the presently selected image of the frame buffer. A Read contains whatever was in the CCD at the time of the read. If the CCD had been sitting for 20 minutes there would 20 minutes of dark noise and light leaks accumulating.

Clear: Clear clears out all the CCD pixels without reading them.

Default: Default sets the CCD read parameters to the default conditions. For a detailed description of these parameters see the FORTH documentation. A default read has 1x1 binning, and all 512x512 pixels are captured by the AFG board.

Obs/Dark: Observe gets its name from the astronomy community and is used to be consistent with Photometrics and Advanced Technologies terminology. **Obs/Dark** pops up a dialog box, that lets you select the destination location, the exposure time, and whether to take an exposure or a dark exposure.
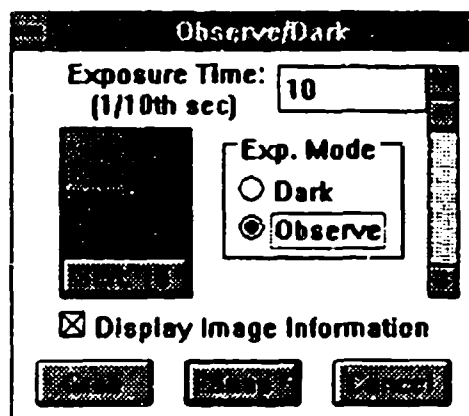
This dialog box stays active throughout the exposure until you exit it. This is convenient for taking many different exposures during a test. The dialog box can be moved wherever you want (even almost all the way off the screen) so that the image being captured is fully visible.

The **Display Image Information** check box indicates whether MIPCTL should store a 'snap-shot' of the camera's status at the time of exposure, store this in the image information buffer associated with this image, and label the image overlay with the information titles. The information parameters are things like time, exposure, filter value, temperatures, location, etc. and are discussed in more detail in Section 6.2. Image information is usually important if you want to archive this image and use it for later analysis. If the image is used purely for temporary information, such as focusing the image, then it is quicker to not display (and store) this information.

Shade: **Shade** produces a horizontal wedge pattern with every 64th pixel increasing in intensity by a value of 1. Thus each row will increase in intensity by 8, and a 512 row image will test out all of the pixel intensities from 0 to 4095. This is useful to test out whether the camera is working, whether the digital input port is working, or even a quick way to test the communications interface.

Setup: **Setup** pops up a dialog box that lets you set details of the CCD read out.

```
┌──────────────────────────────────────┐
│ ▓▓        CCD Settings...          ·  │
│ ┌─Binning──┐  ┌─CCD Gain for HRP──┐  │
│ │  ◉ 1x1   │  │ ┌──────────────────┐│  │
│ │          │  │ │◉ Low: 91e-/ADU   ││  │
│ │  ○ 2x2   │  │ │ Read Noise: 63e- ││  │
│ │          │  │ └──────────────────┘│  │
│ │          │  │ ┌──────────────────┐│  │
│ │ ▓Advanced│  │ │○ Mid: 49e-/ADU   ││  │
│ │          │  │ │ Read Noise: 39e- ││  │
│ │ ┌──────┐ │  │ └──────────────────┘│  │
│ │ │  OK  │ │  │ ┌──────────────────┐│  │
│ │ └──────┘ │  │ │○ High: 25e-/ADU  ││  │
│ └──────────┘  │ │ Read Noise: 24e- ││  │
│               │ └──────────────────┘│  │
│               └───────────────────┘  │
└──────────────────────────────────────┘
```
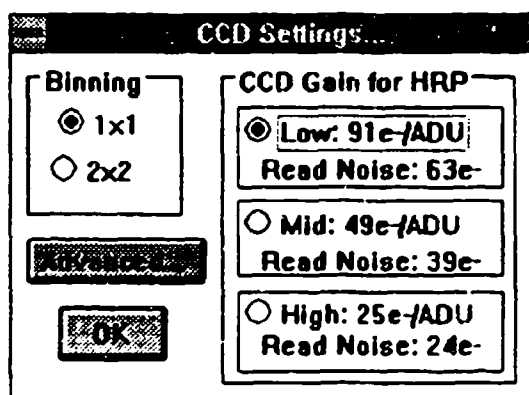
The CCD camera's Gain can be set. The gain is defined as how many electrons it takes to equal one ADU (Analog Digital Unit). The read noise associated with this gain is displayed for your reference. These values are displayed automatically depending on which camera head has been selected in the KEOCCD.INI file. If the heading displays a different camera head, you should correct this by editing KEOCCD.INI with a text editor. The two camera heads currently supported (v5.3.2) are **MIP** and **HRP** (HAARP).

The user can also select the binning factor from this dialog box. Currently, only two standard binning functions are allowed: 1x1 & 2x2 which correspond to the Hi_Resolution and Lo_Resolution images discussed in Section 4.3.3. An inactive button **Advanced** is in the dialog box for the future addition to the software to allow the user to manually set all the individual parameters of the CCD readout. This will allow features such as flexible XY binning & sub-sample readout.

5.3.3 The Control Menu functions

The **Control** menu has the functions that control the HAARP imager control electronics. It looks like:

```
┌───────────────────────────────────────────────────────┐
│ ▓▓           MIP Camera Control              ▼ ▲      │
├───────────────────────────────────────────────────────┤
│ Comm   CCD   ▓Control▓  View  Analyze  Aquisition  Help│
│            ┌─────────────┐                             │
│            │ ▓Control▓   │                             │
│            │ Shutters... │                             │
│            │ Temperature...│                           │
│            └─────────────┘                             │
└───────────────────────────────────────────────────────┘
```
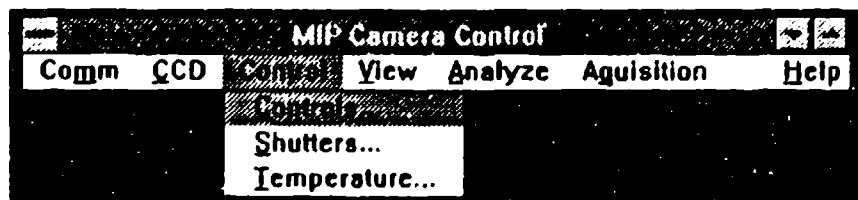
Control: This menu item allows the user to control the filter wheel and the intensifier. The dialog box emulates the control panel on the HAARP imager and is used in much the same way. The target and present values are displayed in two large "LEDs". The user selects a new target for either the intensifier or filterwheel, selects the Intensifier Power On or Off Button, and when ready Sets this new state. If Set is not pushed, this new state will not be updated to the camera and changes will not be reflected in the Preset 'LEDs'.
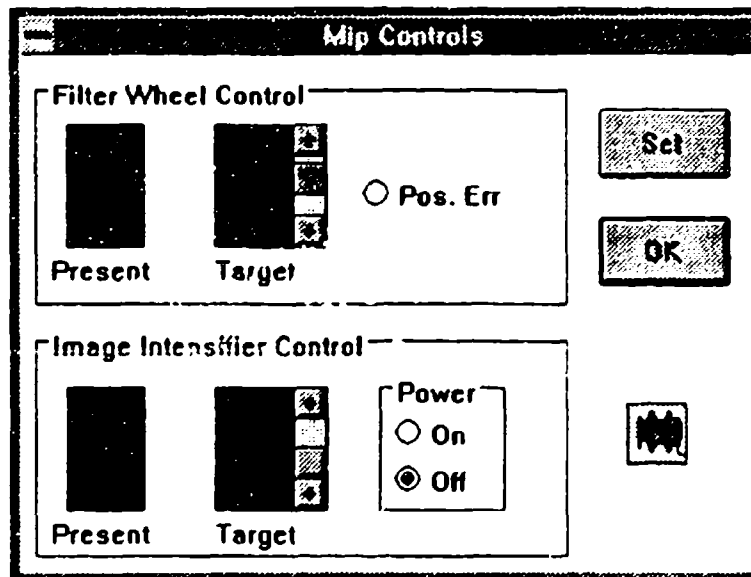
Shutters: The shutters can be manually opened and closed with this menu item. There are status LED's in this dialog that mimic the same function on the HAARP imager control panel indicating whether the shutters are physically open or not. These status lights represent the state of physical microswitches inside the shutters of the camera. This feature is often used to check the shutters for correct operation, or for doing specialized work with the camera (e.g. for visual inspection of the image intesifier: remove the head, open the back shutter, check the light conditions and open the front shutter.)
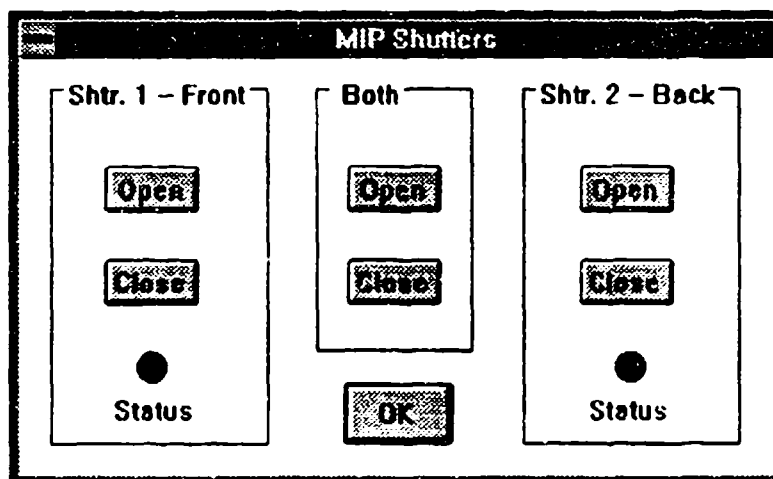
Temperature: This menu item reads all the system temperatures, and displays them in degrees Celsius. It also reads the Intensifier Brightness PhotoDiode. An example of using this and the **Shutter** dialog, would be to open Shutter 1, do a **Temperature**, check the Intensifier Brightness value to see how bright the intensifier is and then close Shutter 1.
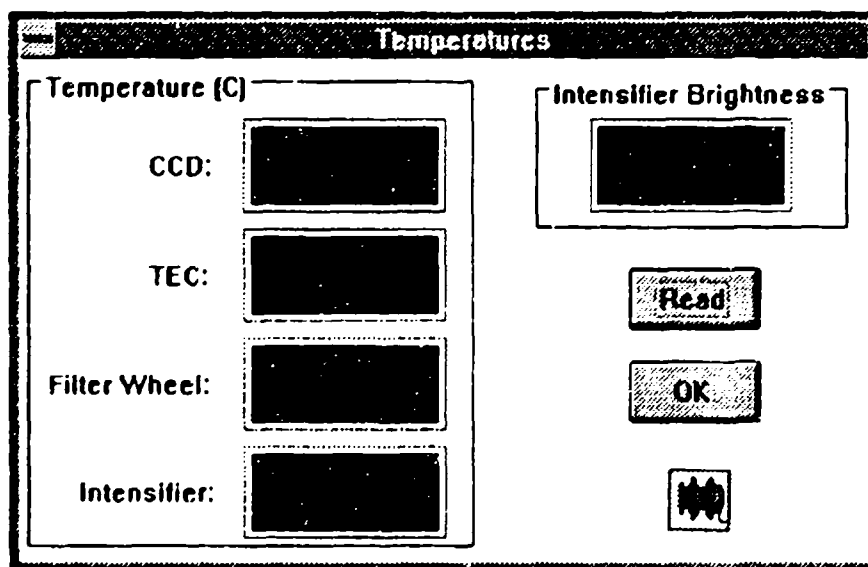
With the current software version (v5.3.2), this dialog box will do a read every 2 seconds and automatically update the 'LED' displays. While reading the camera, the camera icon will come on. Unfortunately, everything is locked up during this period, so hitting the 'OK' button with this icon, will not terminate the dialog box. The user must wait until the icon

105

has returned to the 'ARROW', and then click on the OK button or hit a <cr>. This will be improved in a later software release.

5.3.4 The View Menu functions

The **View** menu controls the basic image processing functions. It looks like:
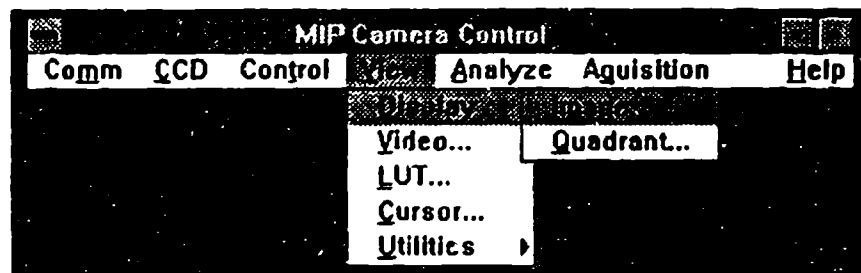
Display: The **Display** dialog lets the user select which image or quadrant to display. Refer to Section 4.3.3 to get a full image and quadrant nomenclature explanation. The user has the option of looking at an image individually or, looking at a whole quadrant.
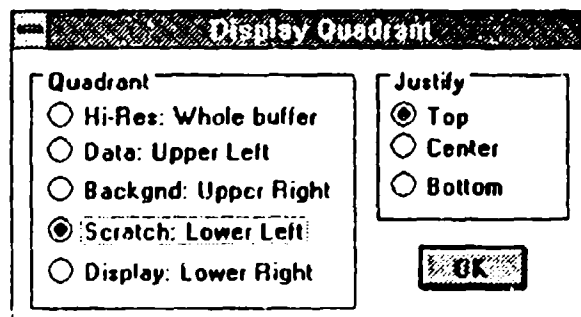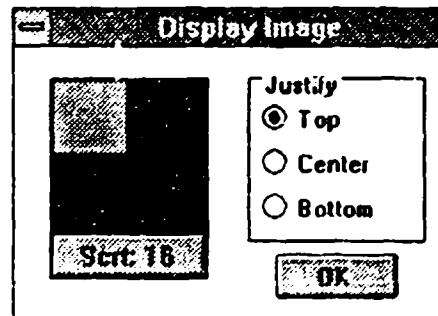
Image or Quandrant displays have three justifications: Because the images have a integral height of 512 pixels, and the VGA screen has a height of 480 pixels, a whole image (or quandrant) cannot be viewed on a VGA display (640x480). The user could change the resolution of the display to SVGA (800x600), but then the screen would have to be recalibrated and the images would be significantly smaller. It was decided that the VGA display would be a better choice, giving larger images.

In order to cover the whole image, the user selects which justification to view the image: Top, Center, and Bottom. **Top** allows the viewer to see the top of the image, cutting of the bottom 32 lines. **Center** 'centers' the image vertically, cutting off the top and bottom 16 lines. **Bottom** displays the bottom of the image cutting of the top 32 lines. The default display mode is **Center**.

A display characteristic of the AFG board should be mentioned here. When displaying the Hi_Resolution **Quadrant**, the AFG board is squeezing a 1024x1024 image area into a 512x480 VGA window. To do this, the AFG board drops every other pixel. This can result in erroneous looking graphics and image artifacts that are not inherent in the image.

Video: This dialog box allows you to change some of the video features associated with the image display. Since the VGA display is mixed with the AFG image display, there are several options that can be used. The default values: VGA Mixing Mode: **VGA Base** and Key Color: **Blue**.

```
┌─────────────────────────────────────┐
│ ═  ░░░░░░░░░Video Options░░░░░░░░░░ │
├─────────────────────────────────────┤
│ ┌─VGA Mixing Mode─┐  ┌─Key Color─┐  │
│ │ ○ VGA Only      │  │ ◉ Blue    │  │
│ │ ◉ VGA Base      │  │ ○ Green   │  │
│ │ ○ IMG Base      │  │ ○ Red     │  │
│ │ ○ IMG Only      │  └───────────┘  │
│ └─────────────────┘  ┌───────────┐  │
│                      │    OK     │  │
│                      └───────────┘  │
└─────────────────────────────────────┘
```
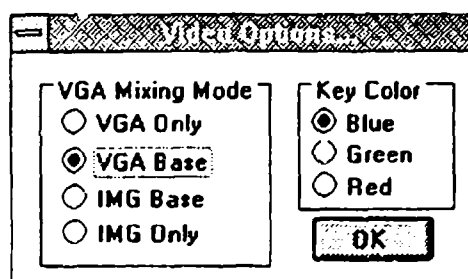
There are four VGA mixing modes: VGA only gives you just the VGA display. This could be useful if you wanted to work just in the Windows environment for a while (say you were developing some new C source code to fix software bugs!). The **MIPCTL** program would still be running, but you would not see any of the frame buffer. *VGA Base*

is the default mixing option. This sets the VGA as the base and anything in the VGA display that matches the selected key color becomes transparent (like the blue windows created by MIPCTL to give a 'port' through to the images in the AFG frame buffer). *IMG Base* is the opposite. The image is the base and anything that is blue in the image will be transparent and show the VGA display. This is not a very practical option but is included more for the sake of completeness. Finally, the *IMG Only* option displays only the frame buffer display and could be useful in some cases. This is terminated by clicking the mouse or by hitting a key. Note that the display will wrap around when you do this as the pixels in a row over 512 will continue into the next pixel of the VGA display and so forth.

LUT: The LUT menu item allows the user to set LUT's on the AFG Board. LUT stands for Look Up Table and represents how a 12 bit pixel value will be converted upon passing through the LUT. The standard uses for an LUT are for the display (as in the first 3 LUT's), or for transforming an image on a pixel-by-pixel basis (such as **Stretch:** mapping all pixels in an image with values of say, 1000 - 3000, to an image with values 0 - 4095).

```
┌──────────────────────────────────────────┐
│ ▬    ▒▒▒▒▒▒▒▒▒▒ LUT Select ▒▒▒▒▒▒▒▒▒▒     │
├──────────────────────────────────────────┤
│ ┌Predefined LUT's ──────────────────┐    │
│ │ ┌─┐                                 │    │
│ │ │ │   ◉  12 Bit Linear              │    │
│ │ └─┘                                 │    │
│ │ ┌─┐                                 │    │
│ │ ║║║   ○  8 Bit Preset               │    │
│ │ └─┘                                 │    │
│ │ ┌─┐                                 │    │
│ │ ║ ║   ○  12 Bit PsuedoColor         │    │
│ │ └─┘                                 │    │
│ └───────────────────────────────────┘    │
│   ┌──┐  ┌──────────────┐   ┌──────┐       │
│   │ ╱│  │  Stretch LUT  │   │  OK  │       │
│   └──┘  └──────────────┘   └──────┘       │
└──────────────────────────────────────────┘
```
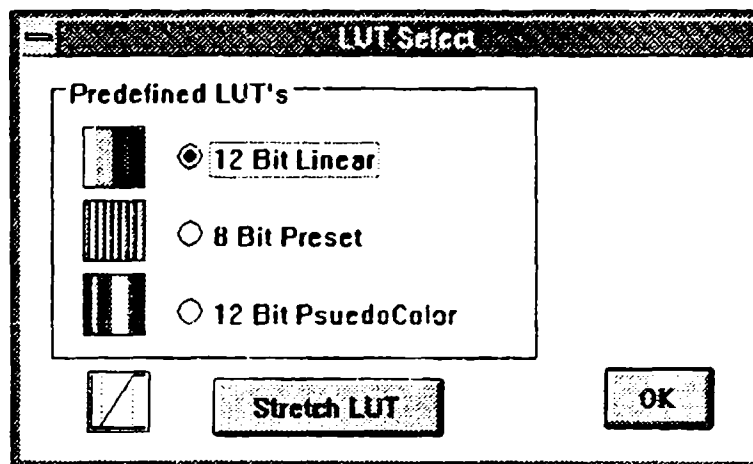
For a detailed description of this it is recommend that you read the Hardware/Software Introductions for the AFG board as well as Chapter 6 of the ITEX Software Manual. In addition, if more complicated LUT operations are needed, they can be implemented using the ITEX LUT commands through the interpreter program of the AFG (INTRP).
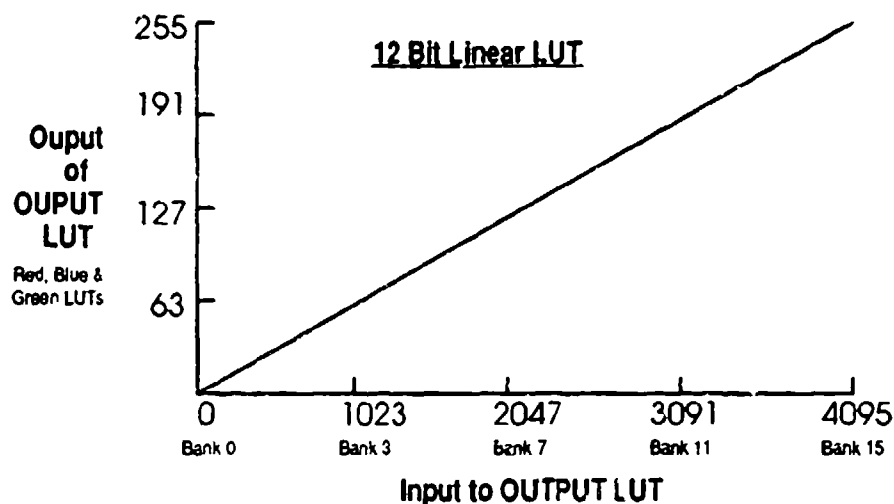
Figure 5.1 -- 12 Bit Linear Grey-Scale Ouptut LUT
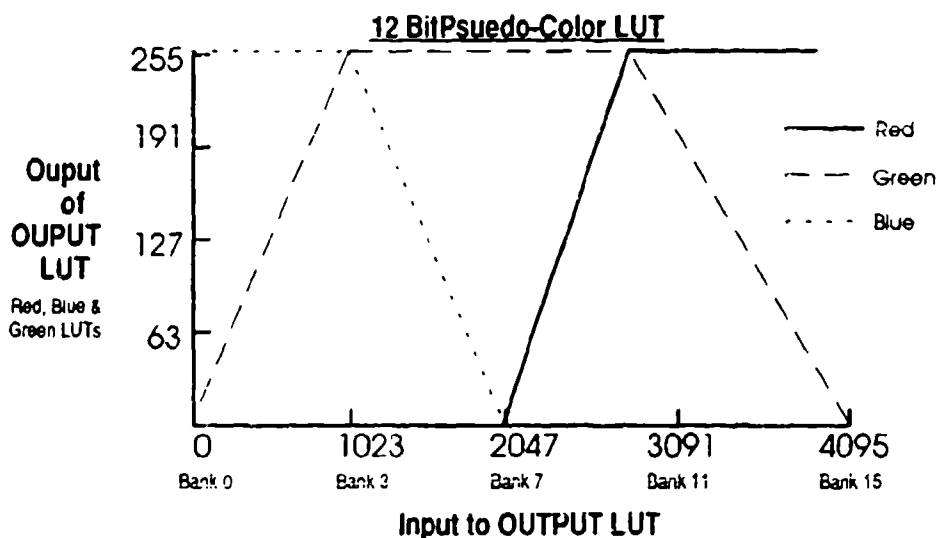
Figure 5.2 -- 12 Bit Psuedo-Color Output LUT

You can just click on the preset LUT's given to you in this menu item to see how different LUT's affect the image. The first three LUT's affect the way the whole frame buffer is displayed (the OUTPUT LUT). There are three default LUTs defined in the HAARP system: 12 bit Linear, 8 bit Default, and 12 bit Psuedo Color. Refer to the ITEX Software manual to get a description of the 8-bit default OUTPUT LUT definitions. Figure 5.1 shows a 12-bit Linear gray-scale LUT, and Figure 5.2 shows a 12-bit psuedo-color LUT. The final LUT affects the INPUT LUT, or the LUT that transforms incoming

109

pixels into the frame buffer. This is used to rescale an image. To do this, you press the **STRETCH LUT** button, and a new dialog appears which allows you to select the source and destination locations (they can be the same, i.e. if you have already saved your data), to enter manually the minimum and maximum values for the LUT, or to use the graphical display to set the LUTs by moving the Min. and Max. arrows.
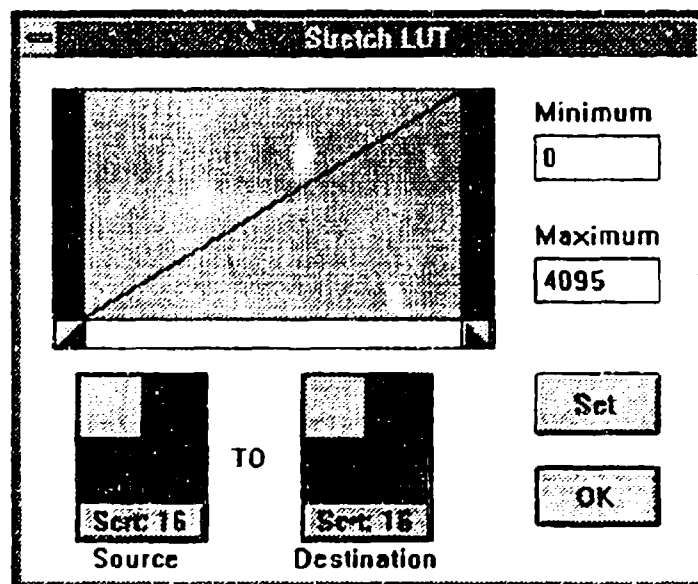
Once you have the desired values, press **SET** and **MIPCTL** will "Snap" the source image into the destination image (passing it through the INPUT LUT).

Cursor: This menu item pops up a display box and changes the cursor to a cross-hair. The display box gives the x,y location of the pixel in the frame buffer's coordinates, and gives the intensity of the pixel at that point. To end the cursor routine, click the mouse once.
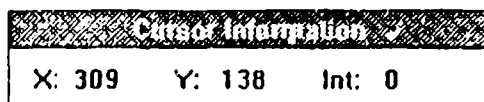
Utilities: The Utilities sub-menu looks like:
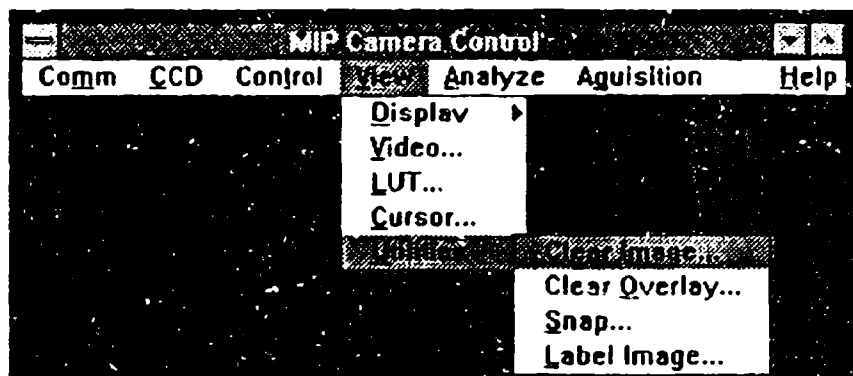
Clear Image: This is a utility to clear different areas of the frame buffer. You can determine what the final pixel value of this area will be by changing the value in the edit box.
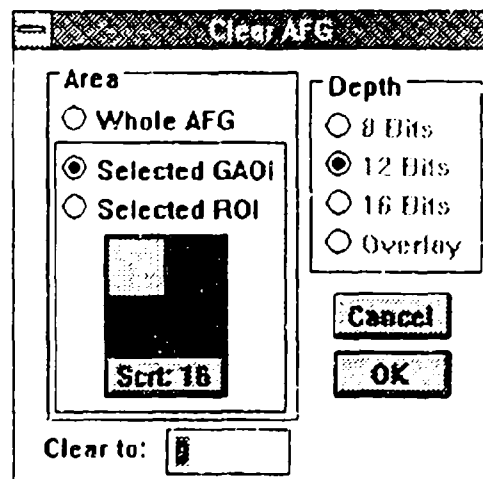
The value will be clipped to whatever destination pixel depth you have selected. You can select areas: The whole frame buffer, a particular image, or the presently defined Region-of-Interest for that image. There are four depths: 8 Bits, 12 Bits, 16 Bits, and the Overlay which is the most significant 4 bits of the frame buffer.

An example of using this utility would be when you wanted to do some 16 bit arithmetic on a 12 bit image that had some graphics data in it. Before doing the arithmetic, you should clear the bits for that region to 0, so that the graphics pixel values do not affect the math. Of course, this would destroy the graphics in the image.

Another example of using this utility, would be if you wanted to do some image arithmetic using a constant operand (the current software version (v5.3.2) does not support constant operand arithmetic). Clear a destination image to the value desired for the constant operand (say 100) and then use this as the destination image of your image arithmetic (discussed in Section 5.3.5). If you were doing addition, your math would become:

Clear Source1 Image to 100
Dest Image = Source2 Image + Source1 Image   (D = S2 + 100)

Clear Overlay: The overlay window is a VGA window that overlays the AFC frame buffer image and is used to draw graphics and text. This insures that none of these graphics objects corrupt the 16 bits of the data stored in the frame buffer. Use this utility to clear these overlay objects:
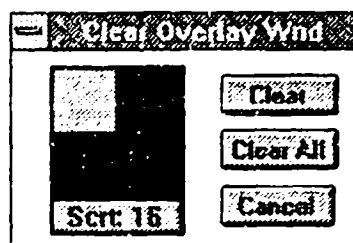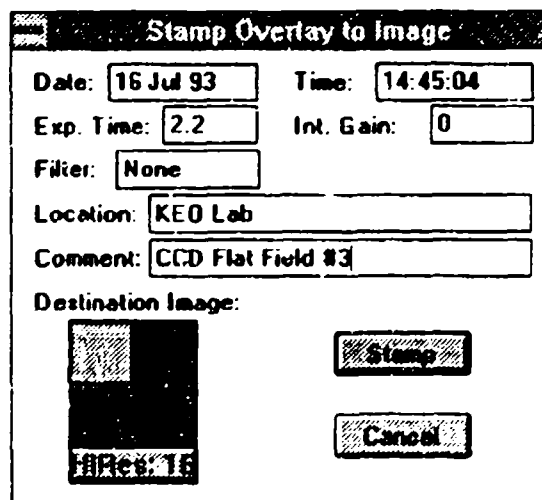
Snap: This utility lets you snap one image to another image. This is done by selecting the source and destination quadrants and then hitting OK (or hitting a <cr>). This is useful for duplicating images in the frame buffer for subsequent manipulation and comparison.
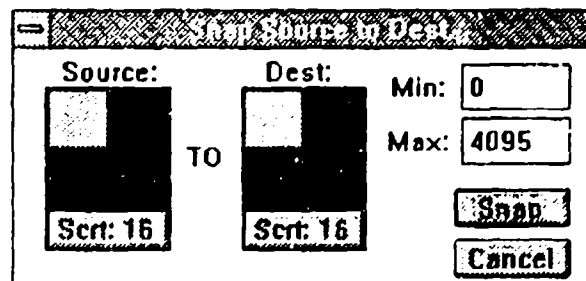
The snap dialog box has provision for setting the Min/Max of the Input LUT. This converts the destination image in exactly the way that **Stretch** does (discussed above under LUT's).

Label Image: The dialog box allows the user to 'stamp' the text fields associated with the image information from the Overlay Window directly into the image pixels. The text will be written into the OVL bits of the frame buffer (B12-B15) and so will not corrupt 12 bit data.

The user can select the destination image. If the image has an image information buffer defined already, this information will be filled into the appropriate edit fields of the dialog box. The user can then modify these values to reflect the final text they want to be stamped into the image. The 'Stamp' button can then be hit and the user will notice the text being transferred directly into the image.

This feature is useful, for example, if the image is to be distributed to another scientist who does not want to spend time deciphering MIPCTL's image header. The text imprinted into the image data will readily identify the image and work on any image processing system that can read 16 bit images.

5.3.5 The Analyze Menu functions

The **Analyze** menu controls the basic image processing functions. It looks like:
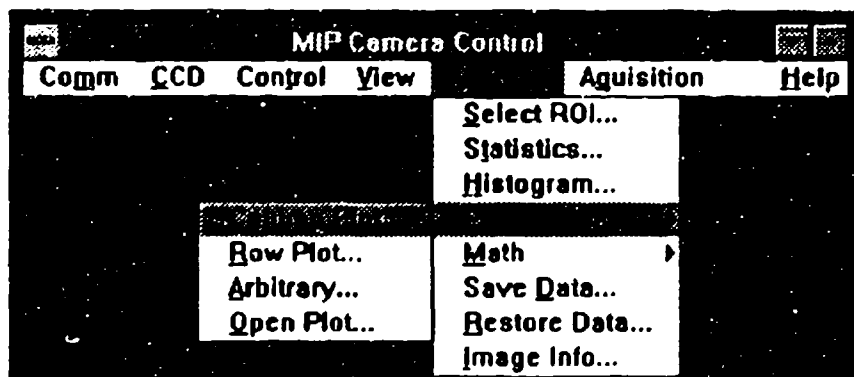
Select ROI: Select ROI (Region-of-Interest) lets you select a ROI either by using the mouse or by entering the coordinates of the ROI from the keyboard.
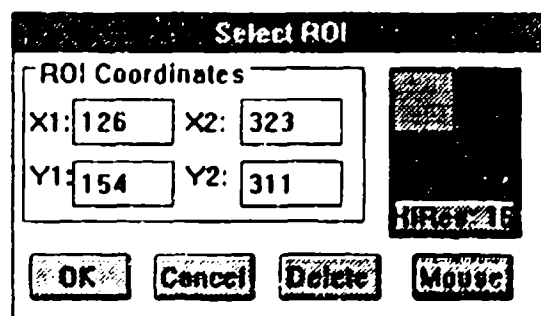
A colored rectangle will appear in the overlay window where the ROI is defined. You can also delete the defined ROI with this menu item. With the current software (v5.3.2), each image can have one ROI presently defined. If the ROI is already defined, it's coordinates will appear in the edit fields when the image is selected.

Once the ROI is defined, it stays in memory until you delete it by hitting the 'Delete' button in the **Select ROI** dialog box. Any subsequent call that uses a ROI, will use this defined ROI. If no ROI is defined, a warning will appear, and you should execute this dialog box to define one.

To use the mouse, click on the mouse button and move the mouse to your desired starting position. Notice that the origin coordinates will be updated with the mouse

movements. Once you are at your desired origin, depress the mouse button, and while holding it down, drag the mouse to you ending point. A colored rectangle will follow the movement of the mouse. If you are dissatisfied with the result, either type in a new ROI, or just press the MOUSE button again and a new ROI will be defined.

Statistics: Statistics calculates the Mean, RMS, Max, and Min of the presently defined ROI in the selected image.
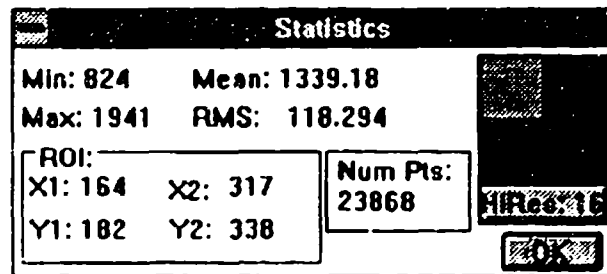
The Statistics dialog box also gives you the coordinates of the ROI and the number of pixels (data points) contained in the ROI. In addition, you can select a new image from within the Statistics dialog and new statistics will be calculated for the defined ROI of this image.

Histogram: Histogram computes a histogram of the selected image (or ROI defined in the image) and plots the results in the standard MIPCTL plotting window.
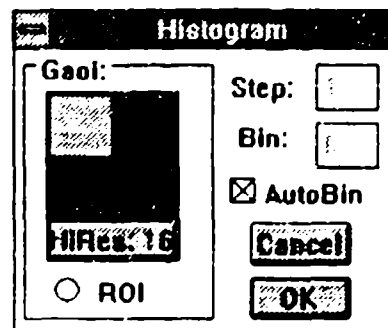
There are two parameters that control the histogram that can be changed. Step size, and Bin size. Step size refers to how many pixels are taken for the sample. 1 corresponds to every pixel, and n corresponds to every nth pixel. The higher the step size, the quicker the histogram will execute, but the less representative of the image the results will be.

Bin Size refers to how many histogram bins are computed. 1 corresponds to every pixel value (0 to 4095 for a 12 bit image), 2 corresponds to 2 pixel values binned into one bin (0 to 2047 for a 12 bit image), etc. This will also greatly increase the speed of histogram execution.

The AutoBin checkbox allows the MIPCTL application to determine the optimum bin size for a plot. Since most Histograms are used for plots, a value similiar to the resolution of the VGA screen would be best. Since the maximum horizontal resolution of our monitors is 1024 pixels, a normal histogram would be 1024 points. The AutoBin calculates this binning factor: For the 16 bit image selected above, a binning factor of 6 is used to map 0 --> 65535 to 0 --> 1023 (65536/1024 = 64 or 2**6).

If higher resolution data for the histogram is desired, then disable the AutoBin mode by clearing the checkbox, and enter the desired binning factor.

Row and Column Plot: Row and Column plot read a line of pixels down a selected column or row and then plots this in the standard Plot window. You can either enter the column or row manually via an edit box, or by using the mouse. A colored line unique to this plotting window appears in the overlay window for the image.
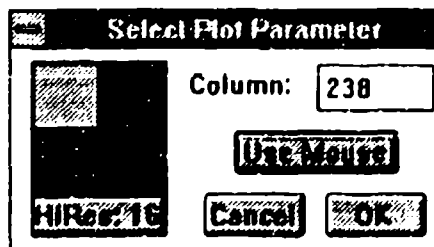
Arbitrary Plot: Arbitrary plot reads a line of pixels across an arbitrary line and then plots this in the standard Plot window. You can either enter the line manually via edit boxes, or by using the mouse. A colored line unique to this plotting window appears in the overlay window for the image. The AFG board reads the pixel values along an approximation of this line. The number of points read correspond to the x distance of this line, not $\sqrt{(x^2 + y^2)}$ as one might expect.
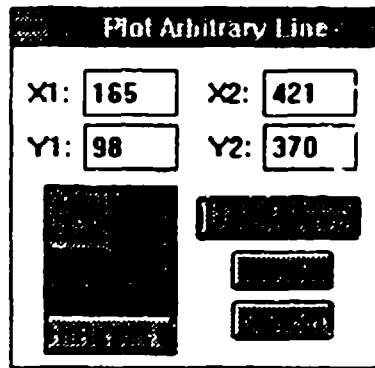
Open Plot: This menu command allows the user to open up a previously saved plot from the MIPCTL application. This plot must be stored in the format discussed below as defined by the MIPCTL application. A user could import data from another application and manually create the necessary header with a standard text editor if needed. Once this plot is opened, it is treated just like any of the other plot windows created with the above commands.

Image Arithmetic from the Analyze menu:
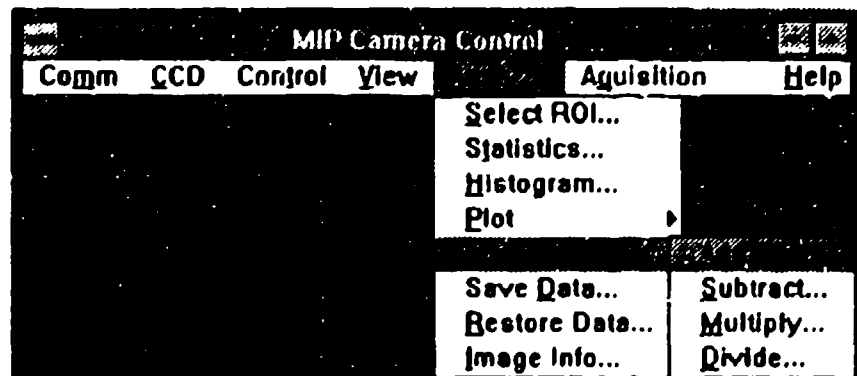
The image math menus pop up a dialog box that lets you select all the necessary features to perform the image arithmetic. Each menu item will bring this dialog up in the default form for it's operation, but you can change any of the displayed parameters. The source and destination GAOI's, the depth, and the post-processing flag are selected.

Image Arithmetic

Scale/Offset — S: [ ]   Off: [ ]

Post-Processing — ○ Unsigned ○ Positive ○ Absolute ○ Signed ○ Negative ⊙ None

OK   Cancel

Source1   Math Op.   Source2   Destination
⊙ + ○ -
○ * ○ /
HiRes: 16   =   HiRes: 16   HiRes: 16
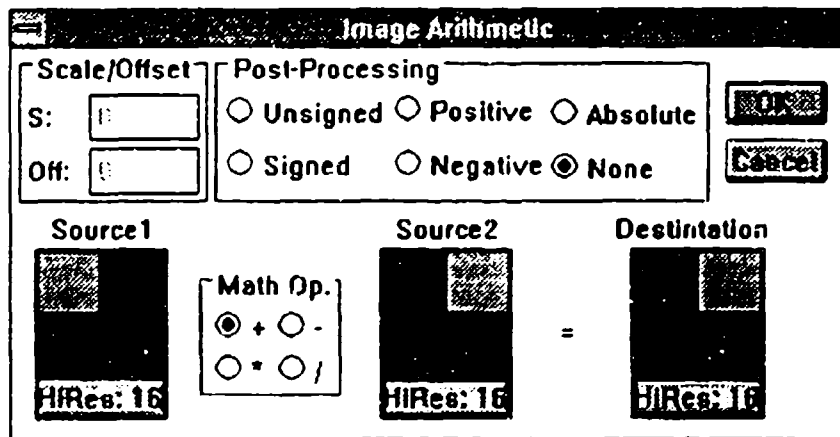
For a detailed description of how the AFG does math, refer to the ITEX Software manual (CH.11) and the ITEX Release Notes v2.2-2, but briefly the math equations are as follows:

Add:	DEST = PFLAG(SOURCE1 + SOURCE2)
Subtract:	DEST = PFLAG(SOURCE1 - SOUCE2)
Multiply:	DEST = PFLAG{ [(SOURCE1 * SOURCE2) >> SCALE]+OFFSET }
⁺Divide:	DEST = PFLAG{ [(SOURCE1 / SOURCE2) >> SCALE]+OFFSET }

where:

PFLAG is the post processing flag and can have the values:

ABSOLUTE: Negative values are converted to positive values and positive values are left unchanged.

NONE : No Post-Processing.

POSITIVE : Positive values are clipped to the maximum GAOI depth and the negative values are set to zero.

NEGATIVE: Negative values are made positive and clipped to 255. Positive values are set to 0.

SIGNED: Numbers are treated as signed values.

SCALE is used in multiply and divide to shift the result of the mult./div. either left or right.

OFFSET is a constant number added to the mult./div. operation after the shift but before the post-processing.

⁺For ITEX v2.2-2, the Divide function was disabled on the AFG board

118

User's Note: When looking at data where differences are of interest, such as calculating the mean/variance curves, negative numbers are significant. Since the math on the AFG board is done in signed arithmetic, you will need to clear all 16 bits of any extraneous information (such as overlays!), and then do signed arithmetic. Even if the images are only 12 bits deep, all 16 bits will be used in the result because of the way signed integers are represented on the AFG board.

It would be a useful exercise for users to experiment with these features with different post-processing options and GAOI depths to make sure they understand the hardware processing on the AFG board. The image math is extremely fast on this board (<16msec), but capabilities are limited by the hardware architecture.

Save Data: Saves the image selected in the Source GAOI control to disk. The image is stored in the ITEX format. The image can be stored on any disk, in any directory and follows the standard IBM CUA guidelines for File Saving.
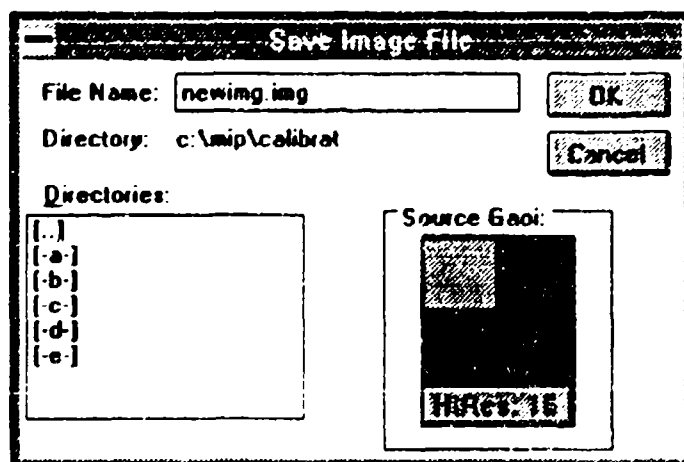
Restore Data: Restore Data allows the user to open up an image from the disk using the standard IBM CUA guidelines for opening files. This image is assumed to be in ITEX format, and will be read into the currently displayed GAOI.
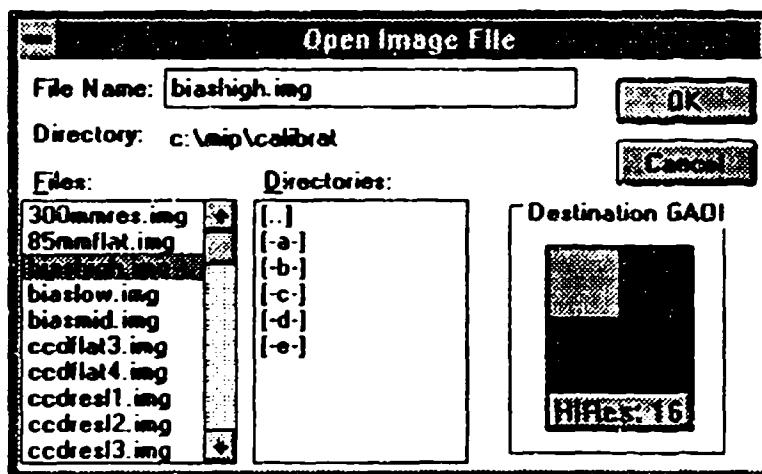
Image Info: This dialog box allows you to quickly look at the image information header of any image on the disk. This box does not initially restore the whole image to a GAOI. If you want to view this image, you can then select the **RESTORE** button and the image will be restored into the selected GAOI.
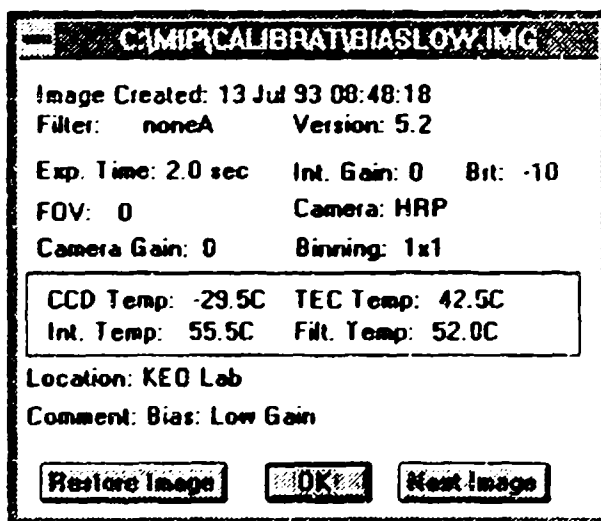
If the image is not an Image Technology image file, this function will not work and an error dialog will appear. The first two bytes of an ITI image file have the characters 'IM' to identify it as an image file. Refer to Section 6.1 for a full explanation of the image file format.

The image information header is looked at next to check that the header was written by the KEO Consultants software package and which software version was used to look at this header. If there is a discrepancy between software versions, or the header was not written

by **MIPCTL**, the header is written into the **Comment:** field of the dialog box so the user can examine it. Refer to Section 6.2 for documentation on the KEO Image Information Header.

5.3.6 MIPCTL Plotting Windows

MIPCTL's Plotting Windows: The MIPCTL has a standardized plotting window feature that allows multiple plots of different types to be created and displayed. These plot windows are standard Windows 3.1 windows and thus can be minimized, scaled, moved, and resized. A plot window has a plot structure associated with it that stores the necessary plot information such as plot type (Histogram, Row, Col, Arbitrary Line), the pixel locations for the plot endpoints (if a line plot), or the StepSize and BinSize (if the plot is a Histogram), the image in the frame buffer that the plot was taken from, and pointers to the actual plot data and the plot window displaying the data.

Many plot windows can be open at the same time. The handling of these plotting windows is done by a Graphics Server that responds both to user commands and messages from the MIPCTL application. While any plot window is open, the Graphics Server will also be open and will appear as an icon in the Window's background. **Do not close the Graphics Server while MIPCTL is still running as it will affect the plots currently displayed by MIPCTL. MIPCTL will open and close the Graphics Server as needed.**
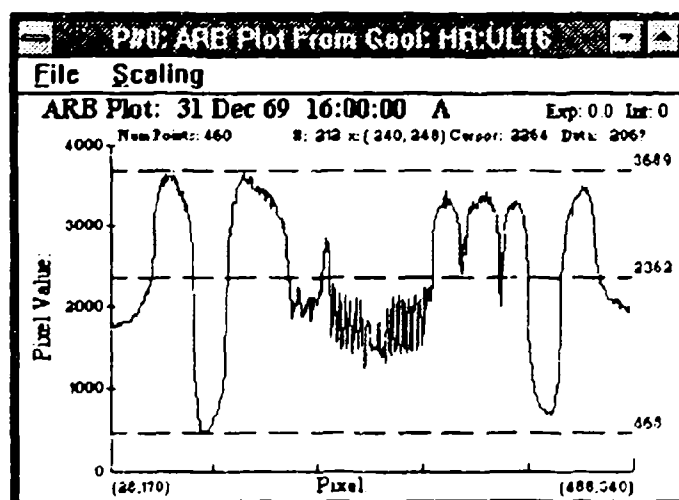
The above plot window shows a plot taken from an **Arbitrary** plot line. The title of the plot window, "P#0: ARB Plot From Gaoi: HR:UL16" tells you which plot window number

121

(0), the kind of plot (ARB), and the image in the frame buffer that the plot was taken from (HR:UL16).

The title of the plot: "ARB Plot: 31 Dec 69 16:00:00   4278A" indentifies the image information of the plot: Type, Date, Time, Filter Wavelength. In addition, the Exposure time and the Intensifier gain are displayed: "Exp: 1.0 Int: 0".

The next status line indicates the number of points in the plot. When the cursor is dragged over the actual plot area in the window, the right hand side of this status line displays information about the cursor position, such as the data # in the plot, the pixel location, the cursor position in Pixel Values, and the data's intensity.

AutoScale & Manual Scale: The plot window initially comes up autoscaling the data. At anytime, if you want to re-autoscale the plot, you can just hit the right-hand mouse button inside the plot window. To manually scale the plot using the mouse, position the cursor on the start point (xmin, ymin) of the plot, depress the left mouse button and drag the cursor to the end point (xmax, ymax) of the plot. As you do this, a yellow window will appear that represents the new plot window to be created as demonstrated in the following figure:
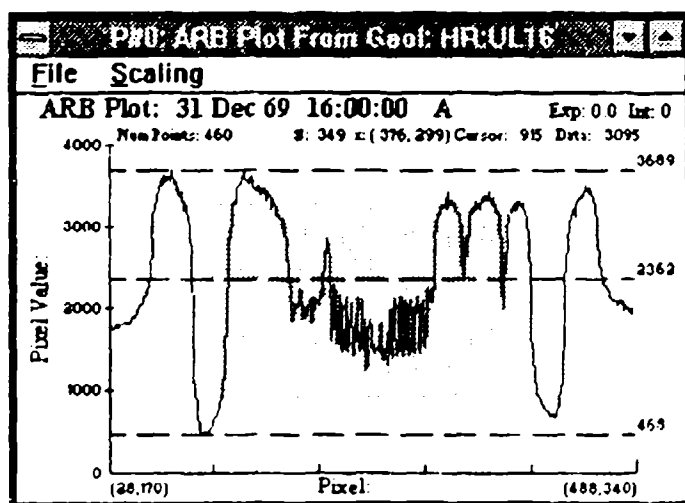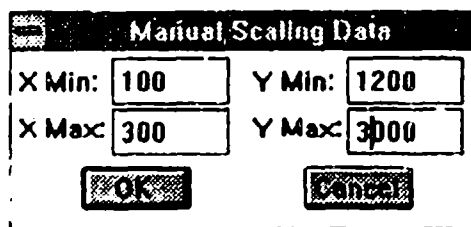
A new window will be created with the re-scaled plot:
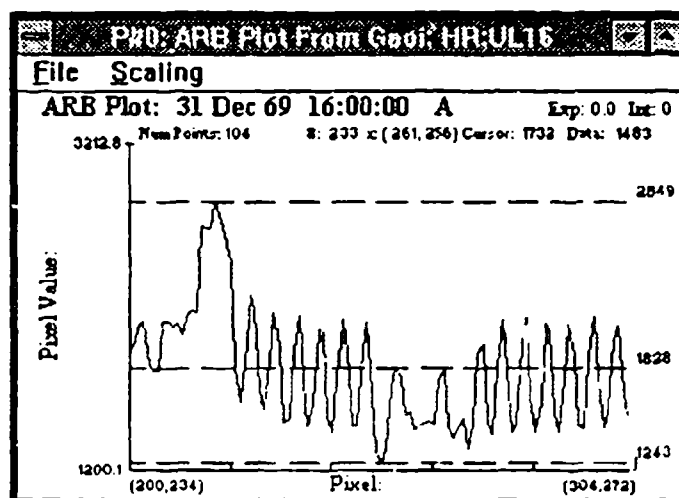
To get back to the **AutoScale** mode, you can simply hit the right mouse button in the plot region again. Alternatively, you can use the mouse commands under the Scaling menu:
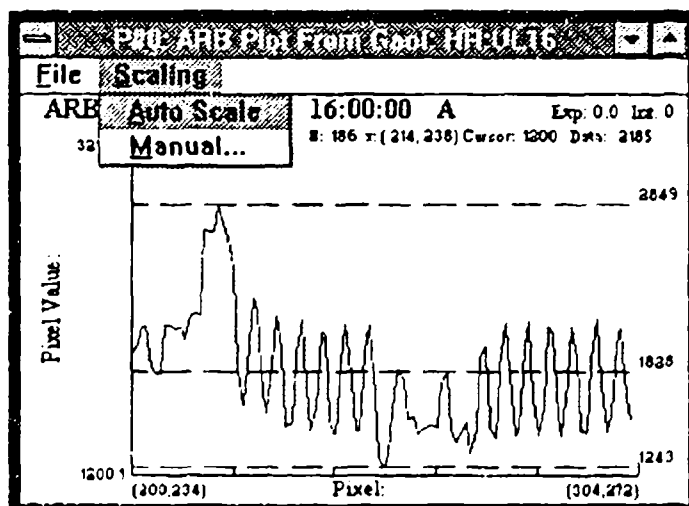
If you choose the Manual scaling menu item, a dialog box will appear that allows you to manually enter the min and max coordinates for each axes. The presently defined limits will appear initially in the edit fields.

Plot Window File Operations:

Under the File Menu in the plot window, you can save a plot, copy it to the Windows clipboard, or print it or exit the plot window and close it. **Closing the MIPCTL app ..tion will automatically close all open plot windows.**
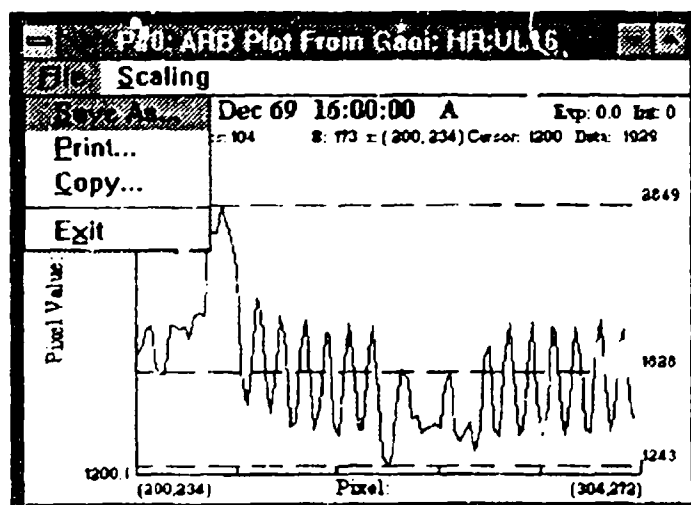
Save As: Saving a plot window prompts for a name of the file and allows you to select the file/disk for the destination file. The MIPCTL standard file extension for a plot file is *.plt. It is recommended that you name all plots with this extension.

The plot is saved by creating an ASCII file that stores the actual data and image information. This file can then be opened from within the MIPCTL application, viewed in any text editor, or opened from within a spreadsheet application like *Microsoft* Excel. Once in this format, complicated charts and analysis features allow the user to view th data. Multiple plots can be created by cutting and pasting spreadsheets. In addition, thi. feature allows an easy portability between users in the scientific community.

124

An example of a plot text file is shown in Figure 5.3.

```
; MIPCTL Plot Output -- Text Format: V1.0 KEO Consultants


Type: PARB '2', Gaoi: HR:UL16 '15', NumPoints: '362'
BegPoint: '71' '129'  EndPoint: '432' '348'


Date: 04 Jul 92  Time: 09:15:34  '76132456'
Filt: '  4560'  Exp: '33'  Int: '0'
Int Brt: '33'  FieldView: '180'
TEMPS - CCD: -26.5C '47'  TEC: 33.5C '166'  Int: 1.5C '103'  Filt: 24.5C '149'
Loc: Ramey Solar Obs
Comment: CRESS Campaign


; Data: Index, Value
0      2704
1      1754
2      1581
3      2025
4      2220
5      2238
6      2179
   ... etc...
```

Figure 5.3 -- Plot Output Text File Example

The first line in the plot is a software identifier. The next two lines contain information about the plot. For the above example, the user could read:

"This plot was an arbitrary line from the image HR:UL1   .   ne buffer. The plot has 362 points and started at point (71,129) and ended at poin,   48)."

The next 6 lines give the image information associated with the image. If there was no information defined, a line would have indicated this. The parameters deliminated with the ' character are the instrument values at the time of the image. For instance the date/time string: 04 Jul 92 09:15:34 is represented as a 4 byte number '76132456' and this value can

be used as the Unix time value. These deliminated values can be extracted by the user for plotting labels, calculations or whatever. For example, an Excel MACRO could be written to extract the temperature values, convert them to Centrigrade and display them in a chart window.

Following the image information, the data occurs sequentially. The first parameter is the data index (or X-axis value), and the second parameter is the intensity data (or Y-axis value). In Excel, these values will automatically appear in individual cells, and thus can be immediately used for charting and calculations.

Printing a plot: Using the **Print** menu command in the plot window, will bring up the printing dialog box of the currently installed printer. A bitmap image of the plot will be scaled to the current page size of the printer (and optionally rotated if the printer driver supports this feature), and sent to the printer via Window's **Print Manager**.

Copying a plot to the Clipboard: Using the **Copy** menu command in the plot window will send an image of the plot to the clipboard. This is then a standard Windows bitmap and can be pasted into any standard Windows application, such as **Word for Windows, Paintbrush,** etc. This is very useful for writing reports and saving images of plots.

5.3.7 MIPCTL Image Acquisition

The next menu item in the MIPCTL application is the **Acquisition** menu and looks like:

The **Setup** menu item gets the user into the Acquisition feature of MIPCTL. This portion of the software is used to acquire data and automate acquisition cycles. When collecting data, this is the main window that will be used.

File   Edit   Operations   Run

```
   Fl  Gaol    Exp  Int  Min  Max  rec dfs
1  1 DA:UL16  10.0  2    0 4095  -   -
2  4 DA:UR16   3.0  2    0 4095  -   -
3  SUB: DA:UL16 - DA:UR16 - DA:UL16
4  PAUSE: 5.0 secs
5  LOOP back 3 entries 4 times
6  ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓
7  SNAP: DA:UR16 to DA:LL16 (200,3000)
```

Filter
- ○ #1 4278
- ○ #2 4865
- ○ #3 5577
- ○ #4 6300
- ◉ #5 7776

Image Buffer

Location: Ramey Solar Obs.
Comment: CRESS Campaign 92
Img Path: c:\mip\calibrat

Exposure: (secs) 5.0

Intensifier Gain: 2

Cycle Time: 30

Display
Min: 100
Max: 4000

☒ Record
☐ Dark Subtract

The acquisition window is divided into three sections. The upper-left hand part of this window contains a list of all the acquisition events. These are actions that are taken during an acquisition cycle. The acquisition cycle executes each event in the acquisition list sequentially. This cycle will repeat itself after the number of seconds determined by the edit field labelled Cycle Time in the lower middle of the window. In the above example, the cycle time is 30 seconds. If the acquisition cycle takes longer than 30 seconds, the cycle will start again with out pausing.

In the lower left hand corner there are three parameters that display what Location and Comment parameters will be stored in the data acquired, and what the path (directory) of the subsequent image data will be. In the above example, the image information headers of the images will have:

Location: Ramey Solar Observatory
Comment: CRRES Campaign 92

The image path will be:

C:\mip\calibrat\

The right hand side of the acquisition window represents the information required to take one image. This is the default 'event' in an acquisition cycle. As we will see, there are several other events that can be defined as well.

The sixth entry in the above acquisition list is highlighted (or selected), so it's acquisition information is displayed in the right hand side of the window. In the acquisition list, the event is represented by the string:

Fi	Gaoi	Exp	Int	min	max	rec	dfs
6	5 DA:UL16	5.0	2	100	4000	rec	--

This could be read as "the sixth aq-event moves to filter #5, takes a Lo-Resolution image using an exposure of 5.0 seconds with the intensifier gain set a 2, and stores the image in the Gaoi: DA:UL16. Once the image is archived (rec), the image is stretched to map it's pixel values of 100 -> 4000 to 0 -> 4095 (thus, increasing the contrast of the image)".

The dfs flag is not set in this aq-event. DFS stands for "dark-frame-subtract" and has been disabled in this version of the software, so the setting of this flag will not currently affect the actions of the aq-event.

This information is mirrored in the right-hand side of the window, where these individual parameters are displayed in a way that allows the user to adjust the values. For instance, in the filter list box, filter #5 is shown to be a 7776Å filter. If the user really wanted this aq-event to use a 5577Å filter, the user would select this filter (#3) by selecting the button in the Filter Group, and this value would be updated in the aq-list.

Perhaps the user would also want to increase the intensifier gain to 3. The user would either edit the number in the Intensifier Gain: box, or change the value by selecting the up-arrow by the side of the box. If the user, wanted to focus on looking at details in the darker part of the image, they could change the maximum mapping value from 4000 to say, 2000, using the same approach. Each time a parameter of the event is changed, the event string in the aq-list will be updated. Thus, after the above changes, the event would look like:

Fi	Gaoi	Exp	Int	min	max	rec	dfs
6	3 DA:UL16	5.0	3	100	2000	rec	--

Individual events can be selected by selecting the event in the aq-list. Once a new event is selected, it's parameters will be updated in the controls on the right-hand-side of the

window. Events can be inserted and deleted using the standard Windows API functions from either the Edit menu, or using the standard keys such as 'Insert' or 'Delete'.

There are other events in the aq-list in the above example that do not come under the category of a default event. There are presently four other types of events defined in the MIPCTL application: SNAP, PAUSE, LOOP and Image Arithmetic. These allow you to perform other operations during an aq-cycle other than just acquiring data and will be discussed in detail in the following pages. As the software evolves, there will be many other operations that the user may want to do during an aq-cycle. Section 6.8 describes in detail all the programming steps necessary to add these additional functions to the acquisition software of MIPCTL.

Acquisition Tables:

Acquisition tables are ASCII text files stored on the hard disk, and called up when the Setup menu item is selected from the MIPCTL menu. The Acquisition table stores all the information pertaining to an acquisition session. When opening up the Acquisition window, MIPCTL looks for the last acquisition table that was used in the acquisition directory. These parameters are saved in the file KEOCCD.INI in directory c:\windows. This file is discussed in more detail in Section 5.3.

The currently opened acquisition table is displayed in the Acquisition Window title. In the above example, the table open was titled CUSP93.AQT. Acquisition tables have the default extension of *.AQT, but this is not required. The '*' at the end of the file name indicates that there have been changes made in the file since it was last opened or saved. If a new file is opened, or this file is closed, MIPCTL will automatically prompt you as to whether or not you want to save these changes.

These tables can be created, opened, modified and saved from within the Acquisition window by using commands in the File menu shown below:
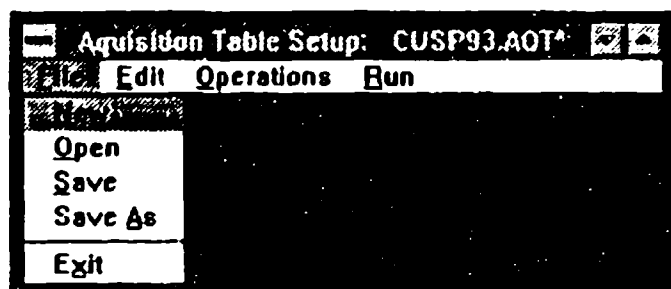
New: Creates a new empty table. The first default aq-event can be added using the currently selected parameters by either modifying the parameters or by typing the 'Insert' key. The new table's title is listed as <New Table> and once it has been modified will be displayed as <New Table>*.

Open: uses the standard Windows API to open an already existing aq-table. The dialog box will display all files with extension *.AQT in the currently defined Acquisition Table directory.

Save: allows you to save the currently active acquisition table. If the name is already defined, the acquisition table is saved and the title is refreshed to reflect it's state. If the table is a <New Table>*, the Save As... dialog function will be executed.

Save As: prompts you for a name of a new acquisition table, and saves it in the currently selected directory. The new name is refreshed into the aq-window title.

Exit: is the standard Windows API command to close the acquisition window. This can also be done by double-clicking on the close box in the upper left hand corner of the window. Exiting the Acquisition window, saves the current acquisition table if necessary and updates the information in KEOCDD.INI to reflect any changes.

Acquisition Table Format:

The acquisition table is stored on disk in a text format and can be viewed with any text editor. A new (or modified) acquisition table could be created this way. The acquisition table defined in the above example looks like:

```
; AQUISITION SETUP INFO v2.1
; Written to:  CUSP93.AQT
;       On:  Tue Jul 27 11:55:53 1993
;
;
Equipment:
Filter 1 [   4278]
Filter 2 [   4865]
Filter 3 [   5577]
Filter 4 [   6300]
Filter 5 [   7776]

Location [SRI Site, Sondestrom]
Comment [CUSP Campaign 93]
Cycle Time [30]
Field of View [180]


;
; Aquisition Table:
; FILT GAOI EXPOS INTEN MIN  MAX  REC? DFS?
; ------------------------------------------
    1   322  10.0    2    0  4095  0   0
    4   326   3.0    2    0  4095  0   0
   -2   326  32.2  322    0     0  5   0
   -6    50   0.0    2    0  4095  0   0
   -7     3   0.4    2    0  4095  0   0
    5   326   5.0    2  100  4000  1   0
   -5   326  33.0    2  200  3000  0   0
```

Figure 5.4 -- Acquisition Table Text File Example

The acquisition table has a header that gives the version number of the acquisition table format software, the acquisition table's file name, the date and the time that the table was written. Next, the equipment information is stored: Filter values, the location and comment strings (discussed above), the Cycle time and the Field-of-View of the lens used during the acquisition cycle. These acquisition tables are useful also for keeping records of what different experiment configurations were used. In the above example, the CUSP experiments in 1993 in Sondestrom at the SRI Site, the filters were configured as 4278, 4865, 5577, 6300, 7776 Angstroms. The cycle time was 30 seconds and the fisheye lens (180 degree Field-of-View [FOV]) was used in the experiments.

Next comes the list of aq-events. These events are a little cryptic as they are translated when read back into the acquisition window. However, from a programmer's point of view, this information could be useful.
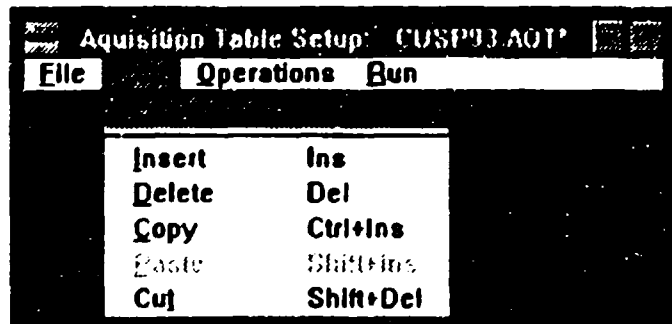
The Edit Menu:

Insert, Delete, Copy, Paste, Cut: The Edit menu allows you to use the standard Windows GUI commands to cut, paste, copy, insert and delete events in the aq-list. These commands work like any standard Window's listbox. To learn more about the functionality of these commands refer to the Windows Documentation (or better yet, experiment with them!).

Equipment: puts up a dialog box that allows the user to change the system parameters and camera configuration.
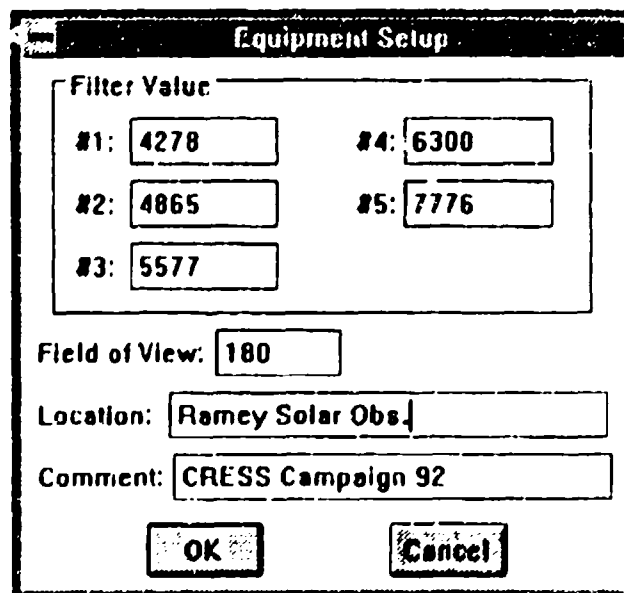
Filter values can be any text up to 8 characters long, but are typically just 4 characters representing the filter wavelength in Angstroms. If you wanted to be more informative, you could, for example, label a filter: 'OI: 6300' where OI is the standard notation for the atomic oxygen line.

The field-of-view edit field (FOV) represents the field of view of the lens currently installed on the camera. Refer to the instrument's manual for the available lenses and their FOV's for the specific instrument. This can be anywhere up to three characters (0 - 180).

The location string is a string up to 25 characters long that records the location of the instrument at the time of the data acquisition. The comment string is a string up to 75 characters long that records any other acquisition specific information that would be helpful for the future identification of the data. A typical comment string would hold the experiment's name such as the CRRES, RODEO or CUSP campaigns.

The Operations Menu:

The Operations Menu allows the different type of operations to be added as entries to the aq-list. These operations are added to the event list by selecting the appropriate menu item from this menu. Once the event has been inserted into the aq-list, it can be modified or viewed by selecting that event from within the aq-list. 'Operation' events can be inserted, deleted, cut and pasted in the same way as any other event in the aq-list.
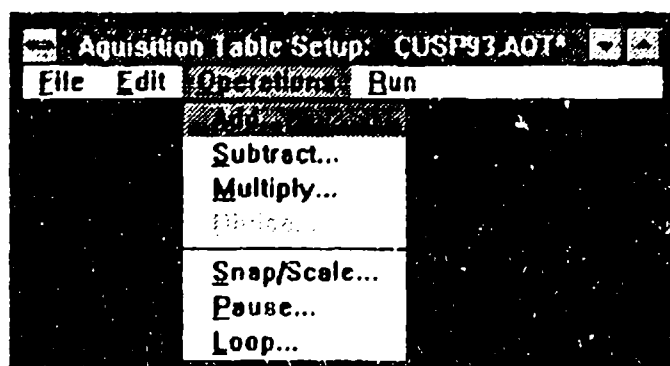
The first four operations that can be added to the aq-list are arithmetic operations. These operations are executed on the AFG board hardware and are therefore done in 'real-time'

(<16msec). The dialog box is exactly the same as the image math dialog box from the Analyze menu discussed in Section 5.3.5. The math is limited however, by the hardware, in the same way as discussed in Section 5.3.5. The present version of Imaging Technology's ITEX-AFG v2.2-2, does not support the hardware divide function. Therefore, as of this software release, a divide during the acquisition cycle is not permitted. Once an arithmetic operation has been entered into the list operation, the event string will be updated to look something like:

5   ADD: DA:LR16 = DA:UL16 + DA:UR16

This represents the operation that will be executed during event #5 of the acquisition cycle. "Add the images in DA:UL16 and DA:UR16 and store the result in DA:LR16".

Snap/Scale: allows an image to be snapped to another GAOI and scaled through the input LUT in the same way the the Stretch LUT operates. The min/max values represent the minimum value that gets mapped to zero, and the maximum value that gets mapped to 4095 (the Input LUT is 12 bits):
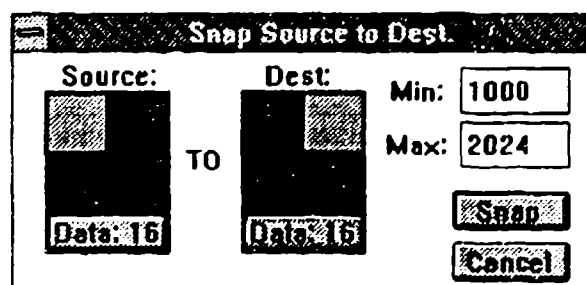
In the above example, the values 1000 to 2024 of the source image DA:UL16 get mapped to 0 to 4095 in the destination image DA:UR16. *The upper 4 bits of the image are unchanged. (Note: the images coming from the cameras are 12 bit images. Normally during acquisition, the upper 4 bits are not used.)* The event string in the aq-list will now appear as:

7   SNAP: DA:UL16 to DA:UR16 (1000,2024)

"The 7th event in the acquisition cycle will snap a copy of the image in DA:UL16 to DA:UR16 remapping the values 1000 --> 2024 to 0 --> 4095 (thus increasing the contrast of the image).

134

Pause: allows the user to let the event wait for a certain amount of time. This is useful for timing purposes and to coordinate inner-loops within the acquisition cycle. The dialog box looks like:
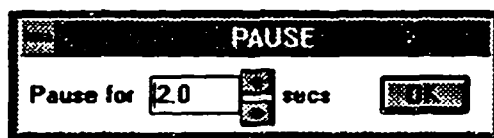
This dialog box tells the event to wait two seconds and then start acting on the next event in the aq-list. The event string would look like:

8   Pause: 2.0 secs

Loop: allows the user to loop back within the aq-list a certain number of times. This allows multiple loops within the acquisition cycle which gives the user the ability to write long complicated acquisition cycles. The dialog box is very straight foward:
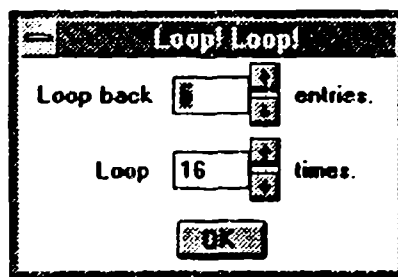
The Loop back feature will only allow you to loop back to the first entry. The entry string will represent this as:

9   LOOP back 5 entries 16 times

The Run menu:

The Run menu allows the user to actually execute an acquisition cycle or an individual event in that cycle.

Table, Entry: starts the execution of the whole acquisition table or the presently selected acquisition entry. When either of these menu items is selected, the acquisition setup window is hidden and a new "Run" window is opened in the upper-right hand corner of the screen:
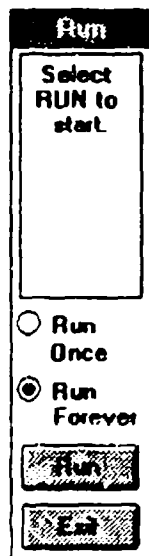
The user can now select whether to run through the table (or entry) once or whether to run continuously, by selecting the appropriate button. Once this is selected, the RUN button is hit to actually start the acquisition session. The RUN button will turn into a PAUSE button to pause the execution. The EXIT button will terminate the acquisition cycle, and return to the acquisition setup window. During execution of the acquisition cycle, status messages will appear in the Run window, letting the user know the current operation in the acquisition cycle.

Set Path: allows the user to select the directory where images are to be archived during the acquisition cycle. Only those data-events with the 'set' flag selected will be recorded to a file in this directory.

Set Directory Path

Set Path for Image Files:
Directory: c:\mip
Directories

- [..]
- [calibrat]
- [dll]
- [document]
- [forth]
- [images]

OK

Cancel
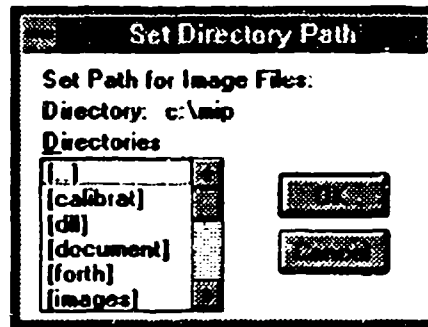
A default file name will be given to images during the acquisition cycle with the format:

YYMMDDHH.NNN where

YY = year

MM = month

DD = day

HH = hour

NNN = the number of the image taken during the hour.

For example, the image 92070409.156 would be the 156th image take during the hour of 0900 on July 4th, 1992. For consistency, all data should stored in UT or Universal Time. To do this, set the computer's clock to UT time from Window's *Control Panel*.

Display Image, Display Quadrant: allows the user to decide whether they would like the currently active image to be displayed soley on the monitor, or whether the whole quadrant that the image is in should be displayed. Selecting this menu item, toggles the option back and forth between Image and Quadrant.

5.3.8 The Help Menu

MIP Camera Control

Comm   CCD   Control   View   Analyze   Aquisition   Help
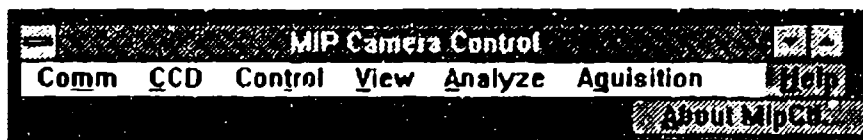
About MIPCtl

About MipCtl: The **About** dialog box is a helpful tool of the **MIPCTL** application. It is dynamically updated to give information on both the **MIPCTL** application's environment and Windows 3.1's environment. The About box displays the current date and time which

137

is useful for checking against a known time source. If the time has drifted or needs to be reset, this can be done via the Windows 3.1 Control panel (in menu MAIN).

```
╔══════════════════════════════════════╗
║          About MipCtl                ║
╠══════════════════════════════════════╣
║  [##]    MIPCTL Application    [##]   ║
║       Tue Jul 27 12:28:45 1993        ║
║  ┌─────────────────────────────────┐  ║
║  │ Version: 5.3.1   Updated: July 22th, 1993 │ ║
║  │ Monitor: 3FGx    Cam: HRP   Port: COM4 │ ║
║  └─────────────────────────────────┘  ║
║  ┌─────────────────────────────────┐  ║
║  │ Win 3.10  DOS 5.0  Free Sys Res: 63% │ ║
║  │ Free LMem: 9282B  Free GMem: 13562KB │ ║
║  └─────────────────────────────────┘  ║
║  KEO Consultants  [##OK##]  Cyril Lance ║
╚══════════════════════════════════════╝
```
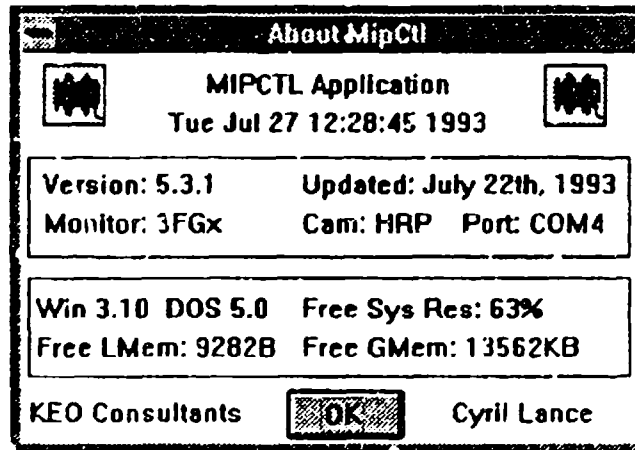
The About box also displays MIPCTL's software version and the date this software was last updated. This is useful for checking image header information and making sure the instrument has the most recent software update.

The monitor, camera head, and COMM port currently selected (and set in KEOCCD.INI) are displayed. In the above case, the software was run using an NEC MultiSync 3FGx monitor, the HAARP camera head, and COM4 for a communications port.

Finally, the About box displays information about the Windows environment. Both the Windows and DOS version numbers are displayed and the percentage of system resources currently available. Next, the amount of available Local Memory and Global Memory are displayed. To fully understand what these parameters are, refer to the Windows documentation. However, these parameters are important to check and make sure that the Windows environment and the MIPCTL application are behaving properly. The above values are typical.

If the **MIPCTL** application starts running slowly, and windows take a long time to draw, something is probably corrupting the Windows environment. To check and see the affect of the **MIPCTL** application, the user can look at the system resources by looking at this dialog box and then close the **MIPCTL** application.

Once the application is closed, the user can check the system resources by using the **About...** dialog box from Window's Program Manager window to see the difference in system resources and hence how much of the system environment **MIPCTL** was using. These features are most useful for debugging in the programming environment, and are not typically needed for the use of **MIPCTL**.

5.4   KEOCCD.INI: System Initialization

The KEOCCD.INI file follows the standard Windows 3.1 API for maintaining application specific initialization information. This file is kept in the windows directory:

C:\Windows\

This file is a standard text file and can be viewed with a text editor just like any Windows 'INI' file (such as WIN.INI and SYSTEM.INI). A typical KEOCCD.INI file is shown in Figure 5.5.

```
[Camera Settings]
Gain=0
Binning=1
Camera=HRP

[System]
Zoom=100
Justify=200
Display=15
Port=COM4
Monitor=3FGx

[CCD Aquisition]
AqtPath=C:\MIP\MIPCTL\
ImgPath=c:\mip\calibrat
LastTable=CUSP93.AQT
```

Figure 5.5 -- KEOCCD.INI Text File Example

The first heading [**Camera Settings**] gives the initial settings of the camera head. **Gain** sets the camera analog gain before it's conversion to digital information This controls the

139

dynamic range and sensitivity of the instrument. Gain values can be 0, 1, or 2 corresponding to HI, MID, and LOW.

Binning can be either 1 or 2 and correspond to the Hi and Lo Resolution images (1x1 binning verses 2x2 binning).

Camera corresponds to which camera head is being used. The current heads available are MIP (MIP) or HRP (HAARP). It is important to keep track of which instrument is collecting data as the flat-field, calibration, and orientations are different for the different instruments.

The second heading [System] sets the MIPCTL application's system parameters and stores the status of the system as it was last used:

Zoom determines which zoom factor was used on the AFG board and sets the current zoom to that value. AFG allows zooming factors of 1, 2, 4, and 1/2. The numbers stored in KEOCCD.INI do not directly correspond to these values but rather to their representation in the program.

Justify determines the justification of display and can be set to top, center and bottom. Again, the numerical values in KEOCCD.INI correspond to the program's representation of these values.

Display sets which image is currently displayed as the top left hand image. In AFG terms, this sets the Pan and Scroll of the display. Again, the numerical values in KEOCCD.INI correspond to the program's representation of these values.

Port determines which valid Windows communications port is used to communicate to the camera. The valid entries are COM1, COM2, COM3, COM4. Currently, the HAARP hardware is setup for the use of COM4.

Monitor determines what monitor is connected to the computer system. This is important as it affects the screen calibration between AFG pixels (not linear in space) with the VGA pixels (see Section 6.5). The two monitors presently defined for the system are MIP's "3D" monitor (the NEC 3D 14" MultiSync) and HAARP's "3FGx" (the NEC 3FGx 15" MultiSync).

The values in KEOCCD.INI are usually maintained by the program itself. The two parameters that should be changed manually using the text editor are the Camera (MIP or HRP) and the Monitor (3D or 3FGx). These parameters can be checked when running **MIPCTL** by using the **About MIPCTL...** menu command discussed in Section 5.3.8.

Occasionally, such as during a system crash, KEOCCD.INI can be corrupted (or even lost). When this happens, use the text editor to correct the corrupted variables such as:

<div align="center">

Monitor = ckjoife   -->   Monitor = 3FGx

</div>

If KEOCCD.INI has been destroyed completely, create a new one in the directory C:\Windows\ and copy the above text from the example KEOCCD.INI tayloring the entries as needed. Then restart the MIPCTL application and check that it is working correctly.

5.5 Notes on AFG Failure

If it appears that there are problems with the AFG board, it may be a software problem with the ITEX drivers. If problems arise, there are several things that you can try to unlock the system. The following is a step through of some things that can unlock the board. You may try all or some of the following.

To see if there is a problem, you can first just exit whatever program you are using to talk to the AFG (such as **MIPCTL** or **INTRP**) and then try to restart it. If the AFG board driver is locked up somehow, you will get an error message during the initialization of the board that will look something like:

<div align="center">

TIGA Error ... 07EH....Couldn't ...

</div>

Exit your program (if it didn't exit automatically), and enter Imaging Technology's board debugger utility DX. If in DOS, go to directory ADGDX and type DX:

<div align="center">

C:> cd \afgdx
C:> dx

</div>

Or if you are in windows, just double click on the DX Icon in the AFG Window.

The DX software will prompt you for a configuration file. Type:

        KEO

after which DX will read this file and set up the board and return a prompt.

Type the following sequence of commands waiting for a prompt before going to the next command:

>init	; Initializes the AFG Board
>extended	; Gives you the extended commands of DX
>test 1	; resets all the registers on the AFG board
>init	
>swrst = 1	; Starts a software reset
>swrst	; Completes a software reset
>exit	

It is usually unnecessary to do a "test 1", but this just makes sure your hardware is running correctly. "SWRST = 1" and "SWRST" must be done together and it executes a software reset on the board.

Once this is done, you can also try and reload the TIGACD TSR to see if that had been corrupted in any way. From DOS, type:

        tigacd -u
        tigacd

Chapter 6   Programming Details

6.1 ITEX Image File format

Images saved to disk by the **MIPCTL** application are stored in the ITEX Image File format. Details are provided in Appendix A of the ITEX-AFG Software Manual provided by Imaging Technology. An outline of the image format is provided here for convenience.

An Image File has the following byte format:

Bytes	Contents
0-1	Characters 'IM' indicate this is an image file
2-3	Comment Length
4-5	Width of image in pixels
6-7	Height of image in pixels
8-9	Coordinates of original X-axis position (horizontal)
10-11	Coordinates of original Y-axis position (vertical)
12-13	File type flag:
	0 -- EIGHT_BIT
	1 -- COMPRESSED
	2 -- SIXTEEN_BIT
14-63	Reserved
64 -- nnn	Comment Area -- variable length; 200 bytes maximum
nnn+1 -- End	Data Area: one byte/pixel (8 Bits) or two bytes/pixel stored in row order, from the top to the bottom of the image.

Figure 6.1 -- ITEX Image File Format

Data is stored in the Intel byte format: Least significant byte first, Most significant byte second. (This is the opposite of 68000 based systems: i.e. ASIPII Multibus, MacIntosh systems.)

6.2 Image Information and the Image Comment

The comment field of the Image File is used to store image information. The image information is a 'snapshot' of the imager when the image was acquired. Images stored using the MIPCTL acquisition system will automatically have image information stamped into the comment field. Other images, created by other applications or by saving an image with no image information structure, will have a nulled comment field. The current software version (v5.3.2) stamps the image information into the comment field with the following format. Refer to Section 6.9 for a chronological version history of the comment field format.

KEO5.3 HRP:22 Jul 93 09:23:48 G[1]W[4278]E[15]V[180]C[63]T[189]I[100]F[140]B[23]
-1-1- Ramey Solar Observatory   CRRES 93
-\127\77\41\95\190"

Figure 6.2 -- Image Information Comment String

This comment string is 200 bytes long including a termination character. The first 6 characters: KEO5.3 indentify this comment as a KEO-created comment from software version 5.3. This software version can then be used to translate the following characters using Section 6.9.

HRP: indentifies the camera head used to acquire the image and next comes a text version of the date and time of acquisition: 22 Jul 93 09:23:48. This information is stamped into the image comment so that anyone dumping the image file, will be able to easily identifiy this information.

Next comes a string of system parameters in the format X[NNN] where X is a one character indentifier for the parameter and enclosed in square brackets is an ASCII representation of the parameter's value. This is not as space efficient as storing the information in binary format, but has the advantage of being easily readable without any software translation (e.g. a user can use a simple file dump). The parameters are taken directly from the image's infomation structure:

G = Intensifier Gain (1 byte)   W = Filter Wavelength (8 bytes)

E = Exposure Time (3 bytes)   V = Field of View (3 bytes)

C = CCD Temp. (3 bytes)   T = TEC Temp. (3 bytes)

I = Intensifier Temp. (3 bytes)   F = Filter Wheel Temp. (3 bytes)

B = Int. Brightness (3 bytes)

The next 5 characters of the string (-1-2-) indicate the values of the Camera Gain (cGain = 1 [Mid-Gain]) and the Binning factor (cBin = 2x2). The next 26 characters are reserved for the Location string (25 char long) with leading spaces (one extra) followed by 76 characters for the Comment string (75 char long) with leading spaces (one extra).

Finally, in the above example are the characters -\127M)_%\0. These are the last 7 characters of the comment string. The first character '-' is a delimiter between the comment string and the binary time variable that follows in the next 5 bytes. The last character is the null character '\0' that terminates the string.

The binary time is encoded into the comment field so that software will readily be able to identify the time using the Unix Time Format supported by C. The first byte is a status byte that indentifies any bytes with value 0 in the four byte time variable. The 0 byte is then set to 127 to ensure that it does not prematurely null the string when it is written to the image file. This is described in more detail in Section 6.9.

6.3 Minimum Files needed to run MIPCTL

This section outlines the minimum files and their directory structure to run the MIPCTL application. This manual does not discuss the necessary files for the Imaging Technology AFG Board and ITEX software. Refer to the ITEX Release Notes v2.2-2 for this information.

MIPCTL files:

```
C:\WINDOWS\KEOCCD.INI             ; Initialization file
C:\WINDOWS\GAOI.DLL               : Gaoi Control DLL file for dialog boxes
C:\WINDOWS\GSW.DLL                ; Graphics Server DLL file for plotting Wins

C:\VISNPLUS\CONFIG\MIP.CAM        ; Camera head definitions for ITEX code
C:\VISNPLUS\CONFIG\KEO.CNF        ; Configuration definitions for ITEX code
C:\VISNPLUS\CONFIG\KEO.MON        ; Monitor definitions for ITEX code
```

Files in the same directory (usually \MIP\MIPCTL):

```
MIPCTL.EXE                    ; Executable usually in \MIP\MIPCTL
MIP1.FNT                      ; TIGA large font: same dir as MIPCTL.EXE
MIP2.FNT                      ; TIGA small font: same dir as MIPCTL.EXE
TEMPTURE.LUT                  ; Temperature LUT -- will be created if none
VGA2AFG.LUT                   ; VGA 2 AFG coordinates LUT -- will be created
                             ; if none
```

6.4 Files and Directory Structure for Program Development

This sections describes all the files needed (as of v5.3.2) for program development of the MIPCTL application and associated DLL's. The directory structure is flexible and can be changed, but care must be taken to make sure that paths are changed in the source code to reflect these changes if the *structure* of the directories is changed.

The MIP development code is in the directory C:\MIP:
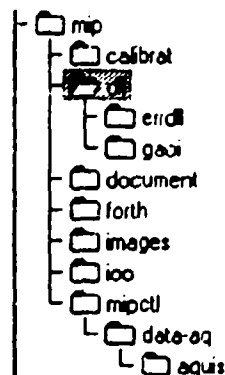
Figure 6.3 -- MIPCTL Directory Structure

The directories in MIP hold the following information:

```
\CALIBRAT              ; Calibration images: not necessary for development
\DLL\ERRDLL            ; An Error Logging DLL: not implemented in v5.3.2
\DLL\GAOI              ; The GAOI Custom Control DLL: Necessary
\DOCUMENT              ; Documentation for HAARP: not necessary
\FORTH                 ; The FORTH and DSP56001 code for the Adv. Tech
                       ; controller board. Not necessary for MIPCTL program
```

	; development, but necessary for development of the control
	; electronics system
\IMAGES	; Images stored on disk: not necessary for development
\IOO	; Image Overlay Object Code: **Necessary**
\MIPCTL	; MIPCTL.EXE development code: **Necessary**
\MIPCTL\DATA-AQ	; Acquisition development code: **Necessary**
\MIP\DATA-AQ\AQUIS	; Acquisition dlg resources: **Necessary**

For the directories that are necessary for program development of the MIPCTL.EXE application, the following files are required:

GAOI DLL Development: development of the GAOI custom control DLL:

Directory: C:\MIP\DLL\GAOI

gaoi.DEF	; Definition file for the DLL executable
gaoi.	; External NMAKE file for building DLL
LibEntry.obj	; DLL entry assembly code
gaoi.DLL	; Final DLL: copied into Windows directory
gaoi.LIB	; Final Library file: copied into c:\mip\mipctl
gaoi.Res	; Compiled resource file for DLL using Resource Compiler
gaoi.c	; DLL C code: File #1
gaoi2.c	; DLL C code: File #2
gaoi.rc	; Resource file
gaoi.h	; DLL function definition hdr file: copied into c:\mip\mipctl
gaoi.dlg	; Custom Control Dialog box
dialog.h	; Dialog box IDD definition header file
quadsel.h	; Quad Select Dialog box IDD definition header file
quadsel.dlg	; Quad Select Dialog box
quadsel.res	; Resource file for dlg box create from Dialog Editor

Image Overlay Object Development:

Directory: C:\MIP\IOO

ioo.c	; Image Overlay Object Code

ioo.h	; Image Overlay Object function definition header file		
ioownd.c	; IOO Window Code		
ioownd.h	; IOO Window function definition header file		

MIPCTL.EXE Program Development: C source code

Directory: C:\MIP\MIPCTL

errors.c	2030	5/4/93	12:02:33pm
filedlg.c	16008	7/15/93	5:55:42pm
mipanlze.c	71440	7/20/93	1:10:10pm
mipccd.c	12073	7/27/93	3:37:12pm
mipcntrl.c	15353	7/23/93	12:00:30pm
mipcomm.c	15206	5/10/93	2:56:04pm
mipdispl.c	41545	7/23/93	11:54:30am
mipfinfo.c	41008	7/16/93	1:31:34pm
mipinit.c	32571	7/23/93	11:58:28am
mipmain.c	5080	5/13/93	9:32:44am
mipmath.c	9511	5/13/93	9:56:20am
mipplot.c	82887	7/15/93	9:30:18am
mipwndpr.c	22235	7/23/93	12:01:52pm

errors.c	; Report errors to a message box handler
filedlg.c	; handles file dialogs for Save/Restore Images
mipanlze.c	; handles functions for code in **Analyze** menu
mipccd.c	; handles functions for code in **CCD** menu
mipcntrl.c	; handles functions for code in **Controls** menu
mipcomm.c	; handles functions for communications to camera head
mipdispl.c	; handles functions for code in **Display** menu
mipfinfo.c	; handles functions for image information and image files
mipinit.c	; handles functions for system initialization
mipmain.c	; WinMain function for MIPCTL.EXE
mipmath.c	; handles functions for image arithmetic
mipplot.c	; handles functions for plotting windows and dlg boxes
mipwndpr.c	; Window procedure for main window

Directory: C:\MIP\MIPCTL\DATA-AQ

aqentry.c	17192	1/3/93	3:59:46pm
aqfiledg.c	11727	5/13/93	12:06:20pm
aqimgmth.c	28534	5/13/93	12:10:40pm
aqrun.c	34755	7/23/93	12:04:52pm
aqsetup.c	42047	6/2/93	9:41:56am
aqtable.c	7323	12/11/92	1:31:52pm
aquis.c	1123	8/7/92	10:50:02am

aqentry.c	; handles functions for the acquisition entry
aqfiledg.c	; handles functions for acquisition table file i/o
aqimgmth.c	; handles functions for image math op's during acquisition
aqrun.c	; handles functions for running acquisition table
aqsetup.c	; handles functions for the setup window
aqtable.c	; handles functions for the table of event entries
aquis.c	; general definitions for acquisition window code

MIPCTL.EXE Program Development: Header Files for C Code

Directory: C:\MIP\MIPCTL

ccdcmds.h	; header for CCD command characters
errids.h	; header for error string ID numbers
library.h	; header for file library #includes
mipctl.h	; function definitions and system #defines
mipextrn.h	; external variable declarations
mipglbis.h	; global variable declarations: used in mipmain.c
miphead.h	; general header file used for most C code
gaoi.h	; header file for GAOI custom control functions

Directory: C:\MIP\MIPCTL\DATA-AQ

aquidds.h	; header file for aqsetup window control idd's
aqsetup.h	; header file for aqsetup window functions
aqtable.h	; header file for aq-table functions
aquis.h	; header file for acquisition structures
aqfile.h	; header file for aq-table file i/o functions

MIPCTL.EXE Program Development: Resource files for dia'og boxes
Resource files created by *Microsoft* C7's Dialog Editor

Directory: C:\MIP\MIPCTL

about.res	; about dialog box
afgclear.res	; clear afg memory dialog box
ccdset.res	; CCD Settings dialog box
clearovl.res	; clear IooWnd graphics dialog box (overlay)
cursor.res	; Cursor dialog box
disimage.res	; Display Image dialog box
FileOpen.res	; Restore Image 'open' dialog box
FileSave.res	; Save Image 'save' dialog box
Histogrm.res	; Get a Historgram dialog box
imaginfo.res	; Image Information dialog box
ImgMath.res	; Image Arithmetic dialog box
labelim.res	; Label Image with ImInfo dialog box
loop.res	; Loop operation (AQ) dialog box
LUTSelct.res	; Select an Output/Input LUT dialog box
ManlInfo.res	; Manual Dimensions for plot windows dialog box
Observe.res	; Observe/Dark dialog box
pause.res	; Pause operation (AQ) dialog box
PlotArb.res	; Get Arbitrary Plot dialog box
PltParam.res	; Get Row/Column Plot dialog box
quadrant.res	; Quadrant select dialog box
SelctROI.res	; Select a ROI dialog box
Settings.res	; Comm Settings dialog box
Shutters.res	; Shutters control dialog box
SnapGaoi.res	; Snap GAOI Utility dialog box
SndMip.res	; Send CCD Command dialog box (shrunk)
Stats.res	; Statistics dialog box
Stretch.res	; Stretch LUT dialog box
Tempture.res	; Temperature display dialog box
video.res	; Video attributes dialog box

Directory: C:\MIP\MIPCTL\D/.TA-AQ\AQUIS

aqentry.res	; AQ -- entry window dialog box (modeless)
aqequip.res	; AQ -- Set Equipment dialog box
aqrun.res	; AQ -- run dialog box
setpath.res	; AQ -- Set Image path dialog box

MIPCTL.EXE Program Development: Hea ier files for dialog boxes
Header files created by *Microsoft* C7's Dialog Editor

Directory: C:\MIP\MIPCTL

about.h	; about dialog box
afgclear.h	; clear afg memory dialog box
ccdset.h	; CCD Settings dialog box
clearovl.h	; clear IooWnd graphics dialog box (overlay)
cursor.h	; Cursor dialog box
disimage.h	; Display Image dialog box
displ.h	; Definitions for display im/quad boxes
filedlg.h	; image save/restore dialog definitions
Histogrm.h	; Get a Historgram dialog box
imaginfo.h	; Image Information dialog box
ImgMath.h	; Image Arithmetic dialog box
labelim.h	; Label Image with ImInfo dialog box
loop.h	; Loop operation (AQ) dialog box
LUTSelct.h	; Select an Output/Input LUT dialog box
ManlInfo.h	; Manual Dimensions for plot windows dialog box
MipCntrl.h	; Filterwheel and Intensifier control dialog box
Observe.h	; Observe/Dark dialog box
pause.h	; Pause operation (AQ) dialog box
PlotArb.h	; Get Arbitrary Plot dialog box
PltParam.h	; Get Row/Column Plot dialog box
quadrant.h	; Quadrant select dialog box
SelctROI.h	; Select a ROI dialog box
Settings.h	; Comm Settings dialog box

Shutters.h	; Shutters control dialog box
SnapGaoi.h	; Snap GAOI Utility dialog box
Stats.h	; Statistics dialog box
Stretch.h	; Stretch LUT dialog box
Tempture.h	; Temperature display dialog box
video.h	; Video attributes dialog box

Directory: C:\MIP\MIPCTL\DATA-AQ

aqentry.h	; AQ -- entry window dialog box (modeless)
\aquis\aqequip.h	; AQ -- Set Equipment dialog box

MIPCTL.EXE Program Development: Dialog files for dialog boxes
Dialog files created by *Microsoft* C7's Dialog Editor

Directory: C:\MIP\MIPCTL

about.dlg	; about dialog box
afgclear.dlg	; clear afg memory dialog box
ccdset.dlg	; CCD Settings dialog box
clearovl.dlg	; clear IooWnd graphics dialog box (overlay)
cursor.dlg	; Cursor dialog box
disimage.dlg	; Display Image dialog box
fileopen.dlg	; image open dialog box
filesave.dlg	; image save dialog box
Histogrm.dlg	; Get a Historgram dialog box
imaginfo.dlg	; Image Information dialog box
ImgMath.dlg	; Image Arithmetic dialog box
labelim.dlg	; Label Image with ImInfo dialog box
loop.dlg	; Loop operation (AQ) dialog box
LUTSelct.dlg	; Select an Output/Input LUT dialog box
ManlInfo.dlg	; Manual Dimensions for plot windows dialog box
MipCntrl.dlg	; Filterwheel and Intensifier control dialog box
Observe.dlg	; Observe/Dark dialog box
pause.dlg	; Pause operation (AQ) dialog box
PlotArb.dlg	; Get Arbitrary Plot dialog box

PltParam.dlg	; Get Row/Column Plot dialog box
quadrant.dlg	; Quadrant select dialog box
SelctROI.dlg	; Select a ROI dialog box
Settings.dlg	; Comm Settings dialog box
Shutters.dlg	; Shutters control dialog box
SnapGaoi.dlg	; Snap GAOI Utility dialog box
SndMip.dlg	; Dialog Box for Sending Command to camera
Stats.dlg	; Statistics dialog box
Stretch.dlg	; Stretch LUT dialog box
Tempture.dlg	; Temperature display dialog box
video.dlg	; Video attributes dialog box

Directory: C:\MIP\MIPCTL\DATA-AQ\AQUIS

aqentry.dlg	; AQ -- entry window dialog box (modeless)
aqequip.dlg	; AQ -- set equipment dialog box
aqrun.dlg	; AQ -- run dialog box
aqtable.dlg	; AQ -- table dialog box (modeless)
fileopen.dlg	; AQ -- open aq-table dialog box
filesave.dlg	; AQ -- save aq-table dialog box
setpath.dlg	; AQ -- set image path dialog box

<u>MIPCTL.EXE Program Development: Resource Files</u>

Directory. C:\MIP\MIPCTL

mnarrdn.bmp	; Min Arrow (down) bitmap for LUTStretch
mnarrup.bmp	; Min Arrow (up) bitmap for LUTStretch
mxarrdn.bmp	; Max Arrow (down) bitmap for LUTStretch
mxarrup.bmp	; Max Arrow (up) bitmap for LUTStretch
camera.cur	; Camera Cursor
crsshair.cur	; Crosshair Cursor
mip1.fnt	; TIGA font for 1x1 binning labeling
mip2.fnt	; TIGA font for 2x2 binning labeling
errids.h	; Header file for error string table
12bitcol.ico	; Icon for 12 bit psuedo-color LUT

12bitlin.ico	; Icon for 12 bit linear monochrome LUT
8bitlin.ico	; Icon for 8 bit linear monochrome LUT
camcntrl.ico	; MIPCTL ICON for application
plotwnd.ico	; Icon for minimized plot window
strchlut.ico	; Icon for LUTStretch Dlg
errstrng.rc	; String table for "Error" messages

MIPCTL.EXE Program Development: Compilation files

Directory: C:\MIP\MIPCTL

Mipctl.def	; Definition file for Windows application
Mipctl.exe	; Final Executable Application: MIPCTL
Gaci.lib	; GAOI control link library for LINKING application
Mipctl.lnk	; Link control file for LINKING application
Mipctl.mak	; NMAKE .mak file for debug version: MIPCTL.EXE
Miprel.mak	; NMAKE .mak file for release version: MIPCTL.EXE
Mipctl.map	; MIPCTL map file of executable file MIPCTL.EXE
Mipextrn.pch	; Precompiled header file for Compile session re-created
	; during a REBUILD from Visual C++
Mipctl.pif	; Windows .PIF file
Mipctl.rc	; Resource File
Mipctl.res	; Compiled resource file using Resource Compiler

Files created by MIPCTL.EXE

Directory: C:\MIP\MIPCTL

*.aqt	; Acquisition table files (ASCII)
*.plt	; Plot Files (ASCII)
*.img	; Images (Binary)
tempture.lut	; Temperature LUT for ADC conversion (ASCII)
vga2afg.lut	; VGA coordinates to AFG coordinates LUT (ASCII)

6.5 Monitor Calibration using the AFG board

The video from the VGA display card in the computer is *mixed* with the video output of the frame grabber card. Thus, the monitor can be thought of as two overlayed images. Depending on the Video mode set in the MIPCTL software (Section 5.3.4), a combination of the two displays will appear at any one time on the monitor. There is often a need to translate from a pixel location in the AFG video to a pixel location in the VGA (computer's) video signal and vice-versa.

An example of translating from AFG pixels to VGA pixles (AFG2Vga) would be to draw a line in the Image Overlay Window of AFG pixel locations x1,y1 to x2,y2 (Ioo'Wnd is in VGA video). In this case, we need to translate the AFG pixel locations to VGA pixel locations and draw at these pixel locations in the VGA screen.

An example of translating from VGA pixels to AFG pixels (Vga2AFG) would be to identify a pixel intensity at the present CURSOR location. Since the cursor is drawn in the VGA video, it's location is given in VGA pixel coordinates. We would need to translate this point in VGA coordinates to it's corresponding point in AFG coordinates and read this value to get the pixel intensity at the CURSOR location.

In order to do this, the monitor must be calibrated with the AFG board. This is done by drawing cross-hairs of known location into the AFG buffer (say every 100 pixels in x and y), and then positioning the cursor over these cross hairs and reading their positions. This calibration is different for the two monitors presently being used by KEO's cameras HRP and MIP. HRP presently uses an NEC 3FGx, and MIP presently uses an NEC 3D monitor. Both of these monitors have been calibrated at VGA resolution (640x480) but are capable of higher-resolutions. If higher-resolutions are desired, then new calibrations should be carried out. When initializing the MIPCTL application, the monitor configuration is looked at, and the appropriate translation calibration is selected.

The calibrations for the MIP imager using the NEC 3D monitor are:

Zoom Factor: x1 $X_{AFG} = 0.924856 * X_{VGA} - 7.745665$ $Y_{AFG} = Y_{VGA}$

Zoom Factor: x2 $X_{AFG} = 0.462174 * X_{VGA} - 4.155732$ $Y_{AFG} = Y_{VGA} / 2$

Zoom Factor: x4 $X_{AFG} = 0.231362 * X_{VGA} - 2.262211$ $Y_{AFG} = Y_{VGA} / 4$

Zoom Factor: x1/2 $X_{AFG} = 1.849 * X_{VGA} - 15.131$ $Y_{AFG} = Y_{VGA} * 2$

The calibrations for the HRP imager using the NEC 3FGx monitor are:

Zoom Factor: x1 $X_{AFG} = 0.94303 * X_{VGA} - 3.5756$ $Y_{AFG} = Y_{VGA}$

Zoom Factor: x2 $X_{AFG} = 0.47131 * X_{VGA} - 1.6803$ $Y_{AFG} = Y_{VGA} / 2$

Zoom Factor: x4 $X_{AFG} = 0.23560 * X_{VGA} - 0.8377$ $Y_{AFG} = Y_{VGA} / 4$

Zoom Factor: x1/2 $X_{AFG} = 0.1.8846 * X_{VGA} - 6.3846$ $Y_{AFG} = Y_{VGA} * 2$

These calibrations are "hard-coded" into the MIPCTL application and have been tested for consistency. Because of the limited resolution, there will be times when the translation between Vga2AFG or AFG2Vga will be one pixel off.

6.6 Commercial Software Installed on the HAARP Imager

This section will discuss the commercial software packages installed on the HAARP imager as delivered with this contract:

<u>Qualitas 386Max Version 7.0</u>   Directory: C:\386max

386Max is a DOS memory optimizer that restructures the DOS memory to allow more available memory space for applications in LO-MEMORY. **386Max** helps programs to run quicker due to less memory swapping, especially DOS programs. It is loaded during the boot-up of the computer and it's affects are transparent to the user. If PIF's need to be created for DOS applications, the PIF editor from Qualitas provides a much more powerful tool than Windows 3.1 PIF Editor.

Installing **386Max** affects CONFIG.SYS, AUTOEXEC.BAT, SYSTEM.INI and WIN.INI.

<u>Berkeley Systems AfterDark for Windows 2.0</u>   Directory: C:\afterdrk

AfterDark is a screen saver that can be used under DOS or Windows 3.1. It is loaded during the bootup of the computer and can be reconfigured, diasabled and enabled via the control panel which is accesed from the mininiized **AfterDark** icon in the lower-left hand corner of the screen. The screen saver helps save the monitor from burn in.

Installing **AfterDark** affects AUTOEXEC.BAT, WIN.INI, and SYSTEM.INI.

<u>**Nu-Mega Bounds-Checker 1.00**</u>   Directory: C:\bchkw

Bounds-Checker for Windows is a debugging tool that checks your application for memory leaks, resource allocation problems, and Windows API parameter violations. An application is run from within BCHKW and a report of errors and other system information such as stack space and memory and resource usage are presented.

Installing **Bounds-Checker** affects SYSTEM.INI.

<u>**Central Point Backup for Windows v7.2**</u>   Directory: C:\cpbackup

Backup for Windows is a backup and restore utility that is used to archive the program development of MIPCTL.EXE. This utility automates the backup of projects. For HAARP backups, we have defined the HAARP-INC and HAARP-FU backups for 'full' or 'incremental' backups. More extensive backups of the hard-disk could be developed using this application.

Installing **Backup** affects SYSTEM.INI.

<u>**Cybernetics CY545**</u>   Directory: C:\cy545

CY545 is a communications DOS utility to talk to a CY545B Stepper Motor Controller chip (the chip that is used to control the Filter Wheel stepper motor). This utility is extremely useful for adjusting the rate parameters and re-programming the CY545's EEPROM. For more information, see the CY545 User's Manual.

<u>**Pinnacle GSW Graphics Server SDK v2.0**</u>   Directory: C:\gsw

GSW is a DLL that is used for plotting and charting routines. This is used by the MIPCTL application for all plotting windows and is crucial to the execution of the plotting features of MIPCTL. The Graphics Server GSW.EXE is launched by MIPCTL whenever there is a plot window opened. This server appears as an iconized window and handles all the messages and drawing of the plot windows. **GSW** has advanced plotting

features that give MIPCTL and the programmer for HAARP a library of routines that facilitate the development of plotting applications.

GSWDLL.DLL and GSW.EXE are installed in c:\windows. GSW.H is installed in c:\msvc\include, and GSWDLL.LIB is installed in c:\msvc\lib.

Lead Technologies LeadTools DLL v3.1   Directory: C:\leaddll

LeadTools DLL is a DLL package that facilitates handling windows with images using all the major formats such as bitmaps, tiff, fitts. The DLL provides window zooming, scrolling, rotation, contrast, brightness as well as archiving and restoring. This package was to be used to develop the handling of HAARP images in a VGA window as well as in the AFG video, to provide more features and capabilities. Lead Technologies also provides this DLL with an upgrade that includes compression capabilities.

LEADDLL.DLL, FIXED.DAT and FIXEDYUV.DAT are installed in c:\windows. L_BITMAP.H, L_ERROR.H, and L_TOOLAP.H are installed in c:\msvc\include, and LEADDLL.LIB is installed in c:\msvc\lib.

Symantec MultiScope Debugger v2.0.1   Directory: C:\mscope

MultiScope Debugger is a sophisticated debugger for the Windows 3.1 operating system. It replaces CodeView for Windows and has enhanced capabilities. MultiScope was used to debug the MIPCTL application before the installation of *Microsoft* VISUAL C/C+ which has it's own embedded debugger. While MSCOPE is much more powerful than the Visual C debugger, it was found not to be necessary for most purposes. MSCOPE has been left on the system in the event that it could become useful for future applications. It was also found that there is a problem running MSCOPE when using an application that uses the Graphics Server GSW. Neither company could find a reason for this incompatibility.

Installing MultiScope Debugger for Windows affects the SYSTEM.INI file.

Microsoft Visual C/C++ v1.00   Directory: C:\msvc

Visual C/C++ is *Microsoft*'s latest C/C++ development environment and is the first truly Windows 3.1 development system from *Microsoft*. The **MIPCTL** application was initially developed using *Microsoft*'s PWB v6.0 and then PWB v7. Both these systems, however, were DOS applications and were not very good and have been deleted from HAARP's hard-drive. Visual C/C++ has been found to be a much easier and robust development environment. *Microsoft*'s C development environments were first chosen because Imaging Technology's ITEX-AFG software require the use of their compilers. ITEX-AFG has now evolved to the point of being a Windows DLL and is not compiler dependent, hence the choice of development environments is more flexible. Symantec and Borland also make good Windows C development environments (the Acquistion software for **MIPCTL** was actually developed under Borland). Visual C/C++ is being used in view of the historical development of MIPCTL.

Installing **Visual C/C++** affects AUTOEXEC.BAT and SYSTEM.INI.

APT ODESSA v1.26   Directory: C:\odessa

ODESSA is the Magneto-Optical Driver for the RICOH MO Drive. The actual driver is MOD.SYS and is loaded in the CONFIG.SYS file. There are several utilities such as MOD.EXE run with the correct switch from M.BAT that allows formatting and partitioning of MO Disks, FASTCOPY.EXE, BFORMAT.EXE and HDD.EXE which allow fast copying, background formatting, and SCSI hard-drive setup. ODESSA is necessary for the use of the MO drive. Once MOD.SYS is installed, the MO Drive is used just like any other installed drive in the DOS/Windows 3.1 operating system.

Installing **ODESSA** affects CONFIG.SYS.

Microsoft Source Profiler   Directory: c\profiler

The **Source Profiler** is supplied with the *Microsoft* C development enviroment to check the performance of a C application.

<u>Imaging Technology ITEX-AFG v2.2-2</u>   Directory: c\visnplus

ITEX-AFG is the core to programming with the AFG Image-Processing board supplied with the HAARP imager. All header files, link libraries, configuration files, debuggers, interpreters, and DLL executables are supplied in this directory. These directories are included in the PATH for proper development. In addition, the TIGA drivers are installed in this directory for communication with the GSP34010 graphics chip on the AFG board. An Interpreter (INTRP) to run ITEX commands without compilation is supplied with this package and is very usefull for controlling the AFG board and for developing new applications.

Installing ITEX-AFG affects AUTOEXEC.BAT.

<u>Microsoft WEP</u>   Directory C:\wep

WEP is a collection of Windows 3.1 games that were delivered with the GATEWAYS 2000 computer. They have been left on the system and take up about 700K of disk space, but can be removed if desired.

<u>Microsoft Windows 3.1</u>   Directory C:\windows

Windows 3.1 is the operating system for the HAARP image processing system. Familiarity with the system is essential for productive use of the system. The software is installed on the HAARP computer in the default configuration and has been optimized for use with the MIPCTL application. Windows 3.1 is started automatically on booting the computer at the end of execution of AUTOEXEC.BAT. If this is not desired, AUTOEXEC.BAT can be modified.

Installing Windows 3.1 affects AUTOEXEC.BAT, and creates WIN.INI and SYSTEM.INI.

<u>Delrina WinFax Pro v3.0</u>   Directory: C:\winfax

WinFax Pro is a Windows 3.1 program to allow FAX sending and receiving. It has a complete selection of cover pages, and a custom cover page for KEO Consultants has been created. In addition, the PhoneBook has been filled with useful numbers relevant to

the development of the HAARP camera. WinFax can send and receive faxes in the background, thus allowing you to continue your work in the foreground. WinFax is started automatically and is set up for COM3 which is the Gateways TelePort Fax/Modem. In some cases, it may not be desired to always have WinFax running. In this case, just close the application using the minimized WinFax icon in the lower left-hand corner of the screen.

Installing **WinFax Pro** affects WIN.INI.

Microsoft Word for Windows 2.0   Directory C:\winword

Word for Windows is a powerful word-processor program and is used for all documentation development. Files with the extension .doc are typically created by Word.

Installing **Word for Windows** affects WIN.INI.

DCA CrossTalk for Windows v2.0   Directory C:\xtalk20

CrossTalk for Windows is a Windows 3.1 communications program and is setup to use the GATEWAYS TelePath Fax/Modem using COM3. This program is useful for electronic communication such as E-Mail and bulletin boards and file-transfers for data, and other information. Several modem configurations are already setup for use that contain commonly called systems.

Installing **CrossTalk for Windows** affects WIN.INI.

6.7 MIPCTL Programming Structures

This section discusses the programming structures used for creating the MIPCTL application, and is a usefull reference for programmers interested in learning more about the software that runs the HAARP imager. This section can be ignored otherwise!

6.7.1 General Area of Interest: GAOI

GAOISTRUCT is used to access GAOI's on the AFG board that are predefined by the MIPCTL program. Using the GAOI structures discussed in Section 4.3.2, three variables

(nQuad, nImage, nDepth) hold the GAOI information while **nIndex** is the actual pointer in the AFG GAOI table used when communicating with the AFG board. *(Defined in MIPGLBLS.H)*

```
typedef struct
    {
    short      nIndex;                    // Index into AFG GAOI Table
    int        nQuad;                     // Quadrant in Frame Buffer
    int        nImage;                    // Image in Quadrant
    int        nDepth;                    // Image Depth
    } GAOISTRUCT;
```

6.7.2 Image Structure: IMAGESTRUCT

IMAGESTRUCT is the main structure used for any image stored in the AFG frame buffer. This structure contains pointers to the GAOI, the Image Overlay Window, and Image Overlay List, ROI, and the Image Information structure. **MIPCTL** creates a table of these structures that correspond to the predefined images discussed in Section 4.3.2. *(Defined in MIPGLBLS.H)*

```
typedef struct                           // IMAGESTRUCT: Image
Structure
    {
    HWND       hWndOwner;                 // Handle to the parent window
    HWND       hIooWnd;                   // Handle to the Ioo window
    short      nGaoi;                     // Ioo GAOI
    HIOO       hIoo;                      // Handle to the image object list
    HIMINFO    hImInfo;                   // Handle to the image info header
    RECT       rROI;                      // Rectangle for ROI
    WORD       flags;                     // Flags for IooWnd
    } IMAGESTRUCT, *pImage;
```

6.7.3 Plot Window Structure: PLOTINFO

The PLOTINFO structure is used to hold all the information necessary to create and maintain a plot window. Plot windows record the plotInfo index number in their window words to access this structure. The PLOTINFO structure for the plot window is filled when the plot data is first read from the AFG frame buffer. *(Defined in MIPGLBLS.H)*

```
struct plot {
    int       type;            // Plot type: PROW, PCOL, PARB, PHIST, PHIST_ROI
    short     nGaoi;           // GAOI data was taken from
    DWORD     rgbClr;          // RGB color of plot
    int       color;           // Plot color for AFG board drawing
    POINT     begPoint;        // Beginning point of plot IN GLOBAL UNITS
    POINT     endPoint;        // Ending point of plot IN GLOBAL UNITS
    long      numPoints;       // Number of points in the plot
    HANDLE    hPlot_Data;      // handle to plot data
    HANDLE    hWnd_Data;       // handle to window data
    HANDLE    hIoo;            // handle to the Ioo for the plot
    HIMINFO   hImInfo;         // Handle to the image info header
} plotInfo[NUM_PLOTS_MAX];
```

6.7.4 Image Information Structure: IMINFOSTRUCT

The IMINFOSTRUCT is used to hold all the pertinent instrument information that was used for the image to which the structure is linked. This information is used to create the image information header that is written into the image file when saved to disk and when image labelling operations are executed. *(Defined in MIPGLBLS.H)*

```
typedef struct {
    long          lImageTime;      // Pointer to a time structure ( 4 Bytes  )
    int           nIntGain;        // Image Intensifier Gain    ( 2 Bytes   )
    unsigned int nExpTime;         // Exposure Time             ( 2 Bytes   )
    char          szFilterVal[9];  // Filter Position           ( 9 Bytes   )
    int           nCCDTemp;        // CCD Temperature           ( 2 Bytes   )
    int           nTECTemp;        // TEC Temperature           ( 2 Bytes   )
```

163

```
    int        nIntTemp;              // Intensifier Temperature  ( 2 Bytes    )
    int        nFiltTemp;             // Filter Wheel Temperature ( 2 Bytes    )
    int        nIntBrt;               // Intensifier Brightness   ( 2 Bytes    )
    short      nFOV;                  // Field of View            ( 2 Bytes    )
    char       *pLocation;            // Location                 ( 2 Bytes    )
    char       *pComment;             // Comment                  ( 2 Bytes    )
    short      nGaoi;                 // GAOI holding the image   ( 2 Bytes    )
} IMINFOSTRUCT;
```

6.7.5 Image Overlay Object Structure: Ioo

The Ioo Structure is a linked-list structure that holds the Image Overlay Object information necessary to maintain an object in the Image Overlay Window. Image Overlay Objects can be text labels, lines, cross-hairs, boxes -- any graphics object that is overlayed on the image. *(Defined in IOO.H)*

```
typedef struct _Ioo {
    IOO_ID    id;                     // Object Type
    HIMAGE    hOwner;                 // Image that owns the object
    HIOO      hNext;                  // Next Ioo in the list
    HIOO      hPrev;                  // Previous Ioo in the list
    WORD      flags;                  // Ioo drawing flags
    char      data[];                 // Ioo data (Object specific)
    } Ioo, far *pIoo;
```

6.7.6 Acquisition Entry Information Structure: aqENTRYINFO

aqENTRYINFO keeps the information necessary to act on the currently selected entry in the acquisition table. *(Defined in AQENTRY.H)*

```
typedef struct {
    aqENTRY         e;                // acquisition Entry information
    aqTABLEENTRY    *inTable;         // pointer to entry in acquistion table
    int             inListBox;        // entry number in List Box
    } aqENTRYINFO
```

6.7.7 Acquisition Table Entry Structure: aqTABLEENTRY

The aqTABLEENTRY structure creates a linked-list of acquisition entries. *(Defined in AQTABLE.H)*

```
typedef struct aqTE {
    struct aqTE  *next;          // Next table-entry in list
    struct aqTE  *prev;          // Last table-entry in list
    aqENTRY   e;                 // Current table-entry
} aqTABLEENTRY
```

6.7.8 Equipment Structure: aqEQUIPMENT

The aqEQUIPMENT structure is used to keep track of equipment settings in the instrument such as which filter is in a certain filter wheel slot. *(Defined in AQUIS.H)*

```
typedef struct {
    int      nDlgItemID:           // Item ID
    char     text[AQ_MAXLABEL];    // String identifying equipment
} aqEQUIPMENT[AQ_NUMEQUIPSTRINGS];
```

6.7.9 Acquisition Entry Structure: aqENTRY

The aqENTRY holds all the information necessary to execute one event in the acquisition table. For a typical acquisition event, parameters such as Gaoi, Intensifier Gain, Exposure are recorded. These parameters can also be used for different things, such as a math operation, where the parameter **nExposure** has a much different meaning. The parameter **nFilter** is used to record which filter is to be used in a data acquisition event and can have the values 1 through 5 for this purpose. If this parameter is less than zero, it flags an alternative type of acquisition event identified by it's absolute value. For example, if **nFilter** is set to -6, this flags that the event is a PAUSE, and the next parameter **nGaoiLoc** is checked to get the pause duration. *(Defined in AQUIS.H)*

```
typedef struct {
    int     nFilter;
    int     nGaoiLoc,
    int     nIntensifier;
    int     nDisplayMin;
    int     nDisplayMax;
    int     bRecord;
    BOOL    bDarkSubtract;
} aqENTRY;
```

6.7.10 Acquisition Desktop Structure: aqDESKTOP

The aqDESKTOP structure holds the present path and filenames for the acquisition information. Acquistion tables are stored in files with the extension *.AQT. MIPCTL keeps track of which acquisition table was last opened using this structure. *(Defined in · QUIS.H)*

```
typedef struct {
    char    szPath[PATHLEN];
    char    szTable[FILENAMELEN];
    char    szEquip[FILENAMELEN];
} aqDESKTOP;
```

6.8 Adding New Operations to the Acquisition Cycle

The acquisition cycle can now accommodate an arbitrary number of operations during an acquisition period in addition to the normal acquisition event. The presently defined operations for the acquisition cycle for MIPCTL v5.3 are as follows:

Acquisition:
    Setup camera -> acquire image -> reccrd(flag) -> label -> rescale

Math:
    Addition, subtraction, multiplication of two images using the
    AFG hardware ALU.

Snap:
> Snap one image to another with the option of scaling image.

Pause:
> Pause for a specified amount of time.

Loop:
> Loop back n entries in the acquisition table j times.

There can be any number of operations that the user may want incorporated into the acquisition cycle. These are now very easy to add, but there are many steps necessary to accomplish this. To facilitate this, follow this procedure:

(1) Dialog Box: There must be some dialog box associated with setting up the parameters of the operations. First create this dialog box defining the controls and dialog name:

Define: NewOp.res   NewOp.dlg   NewOp.h

                           NEWOPBOX   *(Dialog Box Name)*

                           IDC_   *(List of controls in dialog box)*

(2) Add the dialog files to MIPCTL.H In MIPCTL.H, add:

#include NewOp.h   to the list of dialog headers
BOOL FAR aqNewOpEntryDlg(...)   to the list of dialog function declarations.
#define IDM_NEWOP   to the list of menu identifiers for the SETUP
                                                             menu.

(3) Add the dialog resource and menuID to MIPCTL.RC:

Under the SETUPMENU, add:
> MENUITEM "&NewOp",   IDM_NEWOP

Under the list of rcincludes, add:
> rcinclude NEWOP.DLG

(4) Create the dialog function aqNewOpEntryDlg: Write the framework for the dialog function to be used when the dialog box is open. This sets up the input parameters for the acquisition table. This function can be added to AQIMGMTH.C or put in a new source file. If a new source file is used, be sure to add this to the MAKEfile. In addition, this new dialog function must be added to the list of callback functions in the defintinion file: **MIPCTL.DEF**. For examples of doing both of these things, see *aqLoopEntryDlg*. Notice how every operation dialog function makes hMathDlg equal the handle to the operation's dialog in the WM_INITDIALOG message, and sets hMathDlg to NULL when the operation gets a WM_DESTROY message. The name hMathDlg is an unfortunate naming left over from when the only operations were math operations, and should be renamed to hOpDlg in a future version of this software.

(5) Define the command identifier: In AQUIS.H, add a new identier for the NewOp command. All operation identifiers must have a negative value. A zero value is a null entry, and any positive value for the identifier is assumed to be an image acquisition and the identifier is taken as the filter value. See for example:

#define EN_LOOP   -7

(6) Add dialog function declaration to AQSETUP.C and AQRUN.C: Add the function declarations for aqNewOpEntryDlg to the top of these two files. Again, for example, see:

BOOL FAR aqLoopEntryDlg(HWND, WORD, ...);

(7) Define the operation entry variables: An acquisition entry has the structure aqENTRY shown below. The variable names represent what these values are during an image acquisition, but for operations, these values can take on their own meanings.

aqENTRY Structure Definition

```
typedef struct
    {
    int     nFilter;
    int     nGaoiLoc;
```

```
            int     nIntensifier;
            int     nDisplayMin;
            int     nDisplayMax;
            int     bRecord;
            BOOL bDarkSubtract;
            }
        aqENTRY;
```

The only requirement is that e.nFilter hold the operation identifier: EN_NEWOP which must be less than zero to not be interpreted as an image acquisition event. In the LOOP example above, the following definitions were made:

```
nFilter =       EN_LOOP
nGaoiLoc =      The number of entries to loop back to.
nExposure =     The number of times to execute the loop.
```

All other entries are ignored.

(8) Code up the dialog function aqNewOpEntryDlg: Write the code that handles the dialog function messages and sets up the current entry with the correct parameters defined in (7) for the NewOp operation. See *aqLoopEntryDlg* in **AQIMGMTH.C** for an example.

(9) Add the menu option IDM_NEWOP to SetupWndProc in AQSETUP.C: Find *IDM_LOOP* and copy this code, changing *LOOP* to *NEWOP* wherever appropriate. This function opens the dialog box *NEWOPBOX*, when the menu item is selected. After the dialog function is completed, the new entry is added appropriate to the acquisition table. Once added, this entry is treated as any other entry as so all other code is identical.

(10) Add the EN_NEWOP case to 'entry2str' in AQSETUP.C: The function *entry2str* converts the acquisiton table entry passed into a character string to be added to the listbox in the TableDlg. This string should be some English oriented representation of the operation to be acted on. For example, in the EN_LOOP case, the string returned is:

LOOP back 3 entries 5 times

assuming nFilter = EN_LOOP, nGaoiLoc = 3, and nExposure = 5.

(11) Add EN_NEWOP to the EntryDlgProc in AQENTRY.C: The EntryDlgProc must be able to handle creating the operation dialog functions as well. Look for the switch case that looks like:

```
switch ( aqCurEntry.nFilter )
    {
    case EN_ADD:
        .....  break;
    case EN_LOOP:
        ......  break;
    }                                    Etc., etc.
```

Add the case for EN_NEWOP.

(12) Create the DoNewOpEntry() function in AQRUN.C: This is the function that actually does the operation during the running of the acquisition table. This function retrieves the operation parameters from the current aqEntry (tp->e) and does the operation based on these parameters. For example, see:

DoMathEntry(),   DoLoopEntry(),   DoPauseEntry().

Notice how the return values *BEGIN* and *SDONE* are used differently in the operations.

(13) Add the operation to the BEGIN case in AQRUN.C: When the table is running, the BEGIN step is the first step run during an entry sequence. (see function 'nextStepInSequence') If the entry is an operation (tp->e.nFilter < 0), then the appropriate DoNewOpEntry() function that was written in (12) must be called. This case must be added to the switch case here. Roughly:

```
case BEGIN :
        if ( tp->e.nFilter < 0 )          // Is it an operation?
        {
        switch ( tp->e.nFilter )
            {
```

170

```
            case EN_ADD :
              DoMathEntry();
              break;
            case EN_LOOP :
              return DoLoppEntry();
            case EN_NEWOP :
              DoNewOpEntry();        // New operation
              break;

                    .

                    .

                    .

          }
```

These are all the steps necessary for adding a new operation. A full REBUILD must be done before testing out the new code, as the global header file MIPCTL was changed which affect the state of the pre-compiled header.

6.9 Image Comment Chronological Definitions

6.9.1 MIPCTL v2.0: 7/10/92 (CRRES Campaign)

```
szImageComment[0] = 0;                    // Initialize the string


tpInfo = imageInfoHdr[nHdrID].tpImageTime;   // Get time structure
strftime( temp, 20, "%d %b %y", tpInfo );    // Convert Date to string
strcpy( szImageComment, temp );              // copy to image comment
strcat( szImageComment, " " );               // add a space to comment
strftime( temp, 20, "%X", tpInfo );          // Convert Time to string
strcat( szImageComment, temp );              // copy to image comment


sprintf( temp,
       " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-----------%30s %80s",
                    imageInfoHdr[nHdrID].nIntGain,
                    imageInfoHdr[nHdrID].szFilterVal,
                    imageInfoHdr[nHdrID].nExpTime,
                    imageInfoHdr[nHdrID].nFOV,
```

171

```
                                    imageInfoHdr[nHdrID].cCCDTemp,
                                    imageInfoHdr[nHdrID].cTECTemp,
                                    imageInfoHdr[nHdrID].cIntTemp,
                                    imageInfoHdr[nHdrID].cFiltTemp,
                                    imageInfoHdr[nHdrID].cIntBrt2,
                                    szLocation, szComment                    );


    strcat( szImageComment, temp );
```

6.9.2 MIPCTL v4.0.3: 11/19/92

```
    szImage Comment[0] = 0;                        // Initialize the string
// Write KEO Identifier into the header and get software version...
    strcpy( szImageComment, "KEO" );
    sscanf( szVersion, "Version: %s", temp );
    strncat( szImageComment, temp, 3 );
    strcat( szImageComment, " " );
    strncat( szImageComment, szCamera, 3 );
    strcat( szImageComment, ":" );


// Write the Date and Time from the timestamp into the comment...
    tpTime = localtime( &(pINF->lImageTime) );     // Convert to local time


    strftime( temp, 20, "%d %b %y", tpTime );      // Convert Date to string
    strcat( szImageComment, temp );                // copy to image comment
    strcat( szImageComment, " " );                 // add a space to comment


    strftime( temp, 20, "%X", tpTime );            // Convert Time to string
    strcat( szImageComment, temp );                // copy to image comment


    sprintf( temp, " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-%30s %80s",
                                    pINF->nIntGain,
                                    pINF->szFilterVal,
                                    pINF->nExpTime,
                                    pINF->nFOV,
                                    pINF->nCCDTemp,
```

```
                                        pINF->nTECTemp,
                                        pINF->nIntTemp,
                                        pINF->nFiltTemp,
                                        pINF->nIntBrt,
                                        pINF->pLocation, pINF->pComment );


   strcat( szImageComment, temp );
```

6.9.3   MIPCTL v4.1.0: 11/20/92

```
  szImage Comment[0] = 0;                    // Initialize the string
// Write KEO Identifier into the header and get software version...
   strcpy( szImageComment, "KEO" );
   sscanf( szVersion, "Version: %s", temp );
   strncat( szImageComment, temp, 3 );
   strcat( szImageComment, " " );
   strncat( szImageComment, szCamera, 3 );
   strcat( szImageComment, ":" );


// Write the Date and Time from the timestamp into the comment...
   tpTime = localtime( &(pINF->lImageTime) );       // Convert to local time


   strftime( temp, 20, "%d %b %y", tpTime );       // Convert Date to string
   strcat( szImageComment, temp );                 // copy to image comment
   strcat( szImageComment, " " );                  // add a space to comment


   strftime( temp, 20, "%X", tpTime ),             // Convert Time to string
   strcat( szImageComment, temp );                 // copy to image comment


   sprintf( temp, " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-XXXX-%30s %75s",
                                        pINF->nIntGain,
                                        pINF->szFilterVal,
                                        pINF->nExpTime,
                                        pINF->nFOV,
                                        pINF->nCCDTemp,
                                        pINF->nTECTemp,
```

```
                                   pINF->nIntTemp,
                                   pINF->nFiltTemp,
                                   pINF->nIntBrt,
                                   pINF->pLocation, pINF->pComment );


    strcat( szImageComment, temp );
       long *pTime;                              // Pointer to a time variable
    pTime = (long *)(szImageComment + 88);
    *pTime = pINF->lImageTime;            // Put time_t into the header...
    szImageComment[199] = 0;             // Terminate just to be safe...
```

6.9.4 MIPCTL v4.2.1: 12/31/92

```
// Build ImageTechnology image header:    200 Bytes long...


    szInfo[0] = 0;              // Initialize the string


// Write KEO Identifier into the header and get software version...
    strcpy( szInfo, "KEO" );
    sscanf( szVersion, "Version: %s", temp );
    strncat( szInfo, temp, 3 );
    strcat( szInfo, " " );
    strncat( szInfo, szCamera, 3 );
    strcat( szInfo, ":" );


// Write the Date and Time from the time-stamp into the comment...


    tpTime = localtime( &(pINF->lImageTime) );    // Convert to local time


    strftime( temp, 20, "%d %b %y", tpTime );     // Convert Date to string
    strcat( szInfo, temp );                       // copy to image comment
    strcat( szInfo, " " );                        // add a space to comment


    strftime( temp, 20, "%X", tpTime );           // Convert Time to string
    strcat( szInfo, temp );                       // copy to image comment
```

```
// Build the info string:  -XXXX- = ITime (4 Bytes)
    sprintf( temp,
    " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-XXXX-%1d-%1d-%26s %75s",
            pINF->nIntGain,
            pINF->szFilterVal,
            pINF->nExpTime,
            pINF->nFOV,
            pINF->nCCDTemp,
            pINF->nTECTemp,
            pINF->nIntTemp,
            pINF->nFiltTemp,
            pINF->nIntBrt,
              nCGain, nCBin,
            pINF->pLocation, pINF->pComment );


    strcat( szInfo, temp );


// Put binary information into the header...
    pTime = (long *)(szInfo + HDR_TIME);            // Time (4 Bytes)
    *pTime = pINF->lImageTime;


    szInfo[199] = 0;                                          // Terminate just to be safe!
```

6.9.5 MIPCTL v4.2.2: 1/3/93

Header Change: Put the binary time at the end of comment in the event of a 0 byte

```
// Defines for header offsets:  Updated 1/3/92  V4.2.2
..define HDR_GAIN       30
#define HDR_TIME        194
#define HDR_CGAIN       88
#define HDR_CBIN        90
#define HDR_LOCTN       92
#define HDR_COMMNT 119
#define HDR_LOC_LEN  26
#define HDR_COM_LEN 75


// Build ImageTechnology image header:    200 Bytes long...


szInfo[0] = 0;                               // Initialize the string


// Write KEO Identifier into the header and get software version...
strcpy( szInfo, "KEO" );
sscanf( szVersion, "Version: %s", temp );
strncat( szInfo, temp, 3 );
strcat( szInfo, " " );
strncat( szInfo, szCamera, 3 );
strcat( szInfo, ":" );


// Write the Date and Time from the time-stamp into the comment...


tpTime = localtime( &(pINF->lImageTime) );       // Convert to local time


strftime( temp, 20, "%d %b %y", tpTime );        // Convert Date to string
strcat( szInfo, temp );                          // copy to image comment
strcat( szInfo, " " );                           // add a space to comment


strftime( temp, 20, "%X", tpTime );              // Convert Time to string
```

```
strcat( szInfo, temp );                      // copy to image comment


// Build the info string: -XXXX-G-B- XXXX = lTime (4 Bytes), G = nCGain (1 Byte), B = nCBin (1
Byte )


sprintf( temp,
   " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-%1d-%1d-%26s %75s-XXXX",
                        pINF->nIntGain,
                        pINF->szFilterVal,
                        pINF->nExpTime,
                        pINF->nFOV,
                        pINF->nCCDTemp,
                        pINF->nTECTemp,
                        pINF->nIntTemp,
                        pINF->nFiltTemp,
                        pINF->nIntBrt,
                          nCGain, nCBin,                     // Global variables
                        pINF->pLocation, pINF->pComment );


strcat( szInfo, temp );


// Put binary information into the header...
pTime = (long *)(szInfo + HDR_TIME);          // Time (4 Bytes)
*pTime = pINF->lImageTime;


szInfo[199] = 0;                              // Terminate just to be safe!
```

6.9.6   MIPCTL v5.3.0: 7/16/93

Header Change: Encodes the binary time to make sure there are no zero bytes in the
time information that would act as a termination character. Time is now written and
extracted from the information header using the functions (defined in MIPFINFO.C):

```
MakeTimeString( long *pTime, char *szString );
```

lTime = GetTimeFromString(szInfoHdr + HDR_TIME);

The time string is now 5 bytes long. The first being a status byte defined as:

0   1   1   1   SB3   SB2   SB1   SB0

where SB0 --> SB3 are define as 1 if the corresponding byte is non-zero and 0 if the corresponding byte is zero. If a byte in the (long) lTime is zero, this is changed into a 0x7F. So the time string looks like:

SB   B0   B1   B2   B3.

Two examples:

| 0x01020304 ----> | 7F | 04 | 03 | 02 | 01 |
| 0x01020004 ----> | 7D | 04 | 7F | 02 | 01 |

Header definitions (same as 4.2.2):

```
#define HDR_GAIN      30
#define HDR_TIME      194
#define HDR_CGAIN     88
#define HDR_CBIN      90
#define HDR_LOCTN     92
#define HDR_COMMNT    119
#define HDR_LOC_LEN   26
#define HDR_COM_LEN   75
```

// Build ImageTechnology image header: 200 Bytes long...

```
szInfo[0] = 0;              // Initialize the string


// Write KEO Identifier into the header and get software version...


strcpy( szInfo, "KEO" );
sscanf( szVersion, "Version: %s", szComment );
strncat( szInfo, szComment, 3 );
strcat( szInfo, " " );
strncat( szInfo, szCamera, 3 );
strcat( szInfo, ":" );


// Write the Date and Time from the time-stamp into the comment...


tpTime = localtime( &(pINF->lImageTime) );              // Convert to local time


strftime( szComment, 20, "%d %b %y", tpTime ),    // Convert Date to string
strcat( szInfo, szComment );                      // copy to image comment
strcat( szInfo, " " );                            // add a space to comment


strftime( szComment, 20, "%X", tpTime );          // Convert Time to string
strcat( szInfo, szComment );                      // copy to image comment


// Build the info string:


sprintf( szComment,
    " G[%1d]W[%8s]E[%3d]V[%3d]C[%3d]T[%3d]I[%3d]F[%3d]B[%3d]-%1d-%1d-%26s %75s-
XXXX",
        pINF->nIntGain,
        pINF->szFilterVal,
        pINF->nExpTime,
        pINF->nFOV,
        pINF->nCCDTemp,
        pINF->nTECTemp,
```

```
                    pINF->nIntTemp,
                    pINF->nFiltTemp,
                    pINF->nIntBrt,
                    nCGain, nCBin,
                    pINF->pLocation,  pINF->pComment );

strcat( szInfo, szComment );

// Put binary information into the header...

MakeTimeString( &(pINF->lImageTime), szTime );      // Create the time string (char szTime[6])
pChar = szInfo + HDR_TIME;                           // Point to the time string position
strncpy( pChar, szTime, 5 );                         // Copy time string into comment

   szInfo[199] = 0;                                  // Terminate just to be safe!
```

Chapter 7   FORTH and DSP Programming

7.1 Introduction

The heart of the HAARP camera control system is the Controller board supplied by Advanced Technologies. This board has on-board memory with the embedded program that controls all the hardware in the camera. There are two processors on this board: the 68HC11 micorprocessor, and the DSP56001 digitial signal processor. For a more in-depth understanding of the hardware, the user can study the supplied schematics, software listings, and the manuals for the 68HC11 and the DSP56001. These two processors have two distinct responsibilities.

The 68HC11 is the main 'brain' of the camera system. It controls all communication to the outside world via it's RS-422 port. It stores all the command structure to control the hardware aspects of the camera from the KEO control systems such as filter wheel, temperature readout, intensifier gain and shutters, to camera control systems such as CCD read, clear, bias. The 68HC11 is programmed in FORTH and has an embedded FORTH system burnt into it's internal ROM. In addition there is an external EEPROM which permanently stores the user FORTH dictionary and an external RAM for operation. During bootup, the contents of the EEPROM are loaded into RAM. This section discusses the FORTH program burned into EEPROM.

The DSP56001 is a dedicated processor to provide the actual clocking and timing signals required by the CCD. It is used as a state-machine to quickly clock through all the different states that the CCD needs. The DSP code is written in 56000 Assembler and stored in the EEPROM of the 68HC11 as binary executable code. At bootup, the 68HC11 downloads this code to the DSP RAM and tests the DSP's functionality.

The 68HC11 on the Advanced Technologies Control Board is supplied by

> New Micros, Inc.
> 1601 Chalk Hill Rd.
> Dallas, Texas 75212
> (214)339-2204

At power-up, an auto-start routine loads the dictionary from the EEPROM to RAM. loads the DSP code into the DSP RAM, initializes the DSP chip and HAARP/CCD electronics, and starts the HAARP command loop. The HAARP command loop restricts the input to the camera and hence protects the FORTH system from causing catastrophic failures due to operational errors. The HAARP can also be operated as any FORTH system by exiting out of this command loop. At this point, the user can use any of the presently defined words in the FORTH dictionary as well as control all aspects of the MIP camera, add new words, and store a new dictionary in the EEPROM. This gives the user a very powerful ability to control the HAARP, but this facility should be used with extreme caution.

If a problem arises and the dictionary in the EEPROM is corrupted so that you can no longer recover the system, you can jump start the 68HC11 by disabling the chip select pin on the EEPROM and then rebuild the system from the basic FORTH kernel burnt in on the chip's ROM. (See Section 7.7) The FORTH code can then be downloaded from any computer as ASCII text files and then the new dictionary can be stored in the EEPROM. Therefore, it is important to keep the FORTH source files and the DSP binary on whatever computer is being used to controll the HAARP Imager.

The DSP program is also supplied for both reference and for future development. The user can make changes to this code, but this is a very risky step and you should always make sure that you do not change the latest binary (KEO8.DSP as of 11/91) so that you can recover. Once you have changed the DSP assembly language program, you need to assemble it using the DSP assembler and then download this to the FORTH EEPROM using the FORTH word DNLD. Once this is stored in the system's EEPROM, the DSP can be initialized with this new binary by INITializing it.

For more information on the above, please refer to:

MaxFORTH Reference Manual	New Micro
68HC11 Reference Manual	Motorola
56001 DSP Reference Manual	Motorola

7.2 Advanced Technologies FORTH and DSP Code

The Advanced Technologies FORTH Code contains all the necessary words to control the DSP processor, the CCD Camera electronics, and the EEPROM/RAM functions. Normally, you shouldn't have to worry about this code, and will be modifying or adding new dictionary words in the HAARP FORTH code. However, if somehow you have corrupted or need to change the Advanced Technologies code, there are 6 files that need to be loaded. They are (in order of loading): (The 7 corresponds to the version # -- *There might be more recent versions as of this documentation*)

KEOasm7	; Contains 68HC11 assembler words
KEOrom7	; Contains words for EEPROM/RAM control
KEOdsp7	; Contains words for DSP control
KEOadc7	; Contains words for ADC port (PortE)
KEOtim7	; Contains the CCD timing state definitions
KEOcmd7	; Contains the CCD control command words

Each file starts with a FORGET command. This erases the current dictionary from there on. If you made a change in KEOdsp7, you will need to reload everything from KEOdsp7 onwards. After you have loaded the Advanced Technologies FORTH files, you will then need to load the current HAARP FORTH file. For historical reasons, the HAARP FORTH file is called MIPXX.FOR, where XX is the version number. This is because this code was originally developed for the MIP instrument and the programs are identical.

As of 8/93, the version number is 10:   MIP10.FOR

Loading all the FORTH files will have modified the dictionary residing in the RAM and you should immediately be able to test out these changes. However, if you power down the processor, these changes will be lost as the EEPROM wasn't changed.

Once you are sure that your changes work, and these changes have been saved in a set of files with a new version number (i.e. KEOasm9 or MIP11.FOR), you can store this new dictionary in the EEPROM by typing:

STORE
(returns)   DICTIONARY STORED 9465 BYTES

This will take a few minutes and will return with a status message. Once you have stored the dictionary, the autostart sequence must be updated to point to the new dictionary address. To do this type:

MSTART!

(returns)   AUTOSEQUENCE STORED

Now you can turn the power off and your changes will be saved. If something catastrophic has happened and you cannot recover, read Section 7.7 on recovering the EEPROM.

The DSP assembler code is listed as:

KEO8.ASM   ; version 8 as of 11/91

The DSP binary (after assembly) code is listed as:

KEO8.DSP   ; version 8 as of 11/91

If you ever change the DSP code, you will need to download KEO8.DSP as an ASCII file after you have typed the command on the 68HC11:

DNLD

The 68HC11 will capture all the DSP code stored as ASCII characters until the terminating character is sent. Please note that the output file of the DSP assembler needs to modified before being sent to the 68HC11. If you cannot determine the required changes by comparing files, contact Advanced Technologies.

7.3 HAARP FORTH Code

FORTH code for the HAARP was written by KEO Consultants to control the HAARP specific electronics and to provide a quick protected interface to communicate with the host processor.

The KEO Interface Control CCA talks to the 68HC11 through the parallel ports PB and PC, where B is the address, and C is the data. An address demultiplexing GAL on the board deciphers whether data put on the bus is for the KEO Interface. It has been programmed for address hex B400 and is stored in the constant:

```
B400 CONSTANT IOCTL
VARIABLE MIPSTS
VARIABLE MIPCTL
```

The variable MIPSTS stores the last read byte from the KEO Interface CCA, and the variable MIPCTL stores the last written byte to the KEO Interface CCA. These variables are used to mask in new command bits which control specific parts of the HAARP electronics.

Since serial communications is comparably slow, a set of utilities was written to accept characters from the host without echoing. The input command is a character which then gets mapped to a command list numerically using the FORTH :CASE statement. To receive numerical input from the host, the word GETNUM is used. It accepts all characters without echoing until a <cr> is received. GETNUM then converts this st ng into a number and leaves it on the stack.

The HAARP FORTH code's last dictionary words deal with restoring the system on power-up or reset, and defining the auto-start sequence. This is critical code and should not be modified. IT SHOULD ALWAYS BE THE LAST THING LOADED INTO RAM. If any modifications were made to the EEPROM (i.e. by using the STORE command), you must also do a MSTART! to update the autostart address in the EEPROM.

MIP10.FOR contains all the FORTH words necessary for the host to control all HAARP and CCD functions. Upon power-up, the processor is in a loop "MIP" and can only accept the finite commands we allow. However, at any time the host can either restart the processor (COLD boot) or exit the MIP loop to the FORTH kernel mode. At this point any command from the dictionary can be used. New words can be defined and executed as in any FORTH system. Again, these will not be remembered unless you STORE the dictionary in the EEPROM.

7.3.1 MIP Variables

IN_BUFF	; a five character input buffer -- could increase the size if necessary
IOCTL	; a Constant representing the address of the KEO Interface GAL
MIPSTS	; variable holding the last byte read from the KEO Interface CCA
MIPCTL	; variable holding the last byte written to the KEO Interface CCA
ERRLVL	;variable holding the error level for error reporting
ERRSTS	;variable holding a status word for the current errors
NUM_CMD	; a Constant storing the number of MIP commands defined.
	; if you add commands, you need to change this accordingly
MSTART	; a Constant representing the address of the EPROM. Used for the
	; autostart routine

7.3.2 MIP Utility Words

GETNUM

Receives character input until a <cr> has been received, and then converts this to a number leaving the value on the stack. *Note: Could add a timeout functionality to this, in case a <cr> is never received.*

TERM_EMIT

Emits a "#". This is used as a terminating character notification to the host.

CGAIN

Uses the :CASE dictionary event to execute either the HI-GAIN, MID-GAIN, or LO-GAIN commands based on the number on the stack (0-2). Does no error checking, so this MUST be done before CGAIN is called. The words executed set the CCD camera gain for the ADC conversion.

CTL!

Stores the byte on the stack in MIPCTL and outputs this to the device at location IOCTL (in this case the KEO Interface CCA).

CTL@

Reads the byte at location IOCTL (KEO Interface CCA) and stores the value in MIPSTS and also leaves it on the stack.

186

ERR!   OR's in the present error bit on the stack into the error status
                  word ERRSTS.

ERRCHK   (<cond> <err#> **ERRCHK** <continue flag>)
                  ERRCHK looks at a condition <cond>left on the stack, and
                  sets the error bit determined by <err#> into the error status
                  word depending on the ERRLVL set for the system.

SHTR_STS   Gets the status bits of the shutters on leaves them on the
                  stack as B0 and B1.

CLS   Closes both shutters with no status check. Does not modify
                  the stack at all, and does not update MIPCTL or MIPSTS.

OPS   Opens both shutters with no status check. Does not modify
                  the stack at all, and does not update MIPCTL or MIPSTS.

7.3.3 Error Checking System

An error reporting scheme has been implemented on the HAARP imager. There
are three levels of error reporting that can occur:

        0   No error reporting, continue operation
        1   Report error, but continue operation
        2   Report error, abort operation

The Utility word ERRCHK is used to implement this feature. The error reporting is
determined by the error level stored in the variable ERRLVL. This error level is set by the
host dynamically. The default error level is always 2 (report and abort). Functions that
return errors report all errors as negative numbers.

7.3.4 MIP Command Words

Note: All the command words follow the MIP command format and finish by emitting the terminating character "#". None of these commands modify the stack in any way.

RD_SHTRS   Reads the shutter status, pops this value off the stack and emits the terminating character.

OPEN_SH1   Opens Shutter 1, waits 100 msec, reads the shutter status, and returns 0 if open, and -1 if closed (error), and finally emits the terminating character.

OPEN_SH2   Opens Shutter 2, waits 100 msec, reads the shutter status, and returns 0 if open, -2 if closed (error), and finally emits the terminating character.

CLOSE_SH1   Closes Shutter 1, waits 100 msec, reads the shutter status and returns 0 if closed, -1 if open (error), and finally emits the terminating character.

CLOSE_SH2   Closes Shutter 2, waits 100 msec, reads the shutter status and returns 0 if closed, -2 if open (error), and finally emits the terminating character.

OPEN_SHTS   Opens both shutters simultaneously, waits 100 msec, reads the shutter status, and returns 0 if both are open, 6 if shutter 1 is closed (error), 5 if shutter 2 is closed (error), 7 if both are closed, (error), and then emits the terminating character.

CLOSE_SHTS   Closes both shutters simultaneously, waits 100 msec, reads the shutter status, and returns 0 if both are closed, 6 if shutter 1 is open (error), 5 if shutter 2 is open (error), 7 if both are open (error), and then emits the terminating character.

CCD_BIAS   Closes the shutters (with no status check), clears the CCD
three times (to scrub out any residual charge), reads
the CCD and finally emits the terminating character. Does
not modify the stack in any way.

2BIN   Sets the CCD to read out in 2x2 binning with 256x256
serial and parallel lengths. Emits a terminating character.

1BIN   Sets the CCD to read out in 1x1 binning with 512x512
serial and parallel lengths. Emits a terminating character.

CCD_GAIN   Gets the Gain selection from the host, checks for >2
condition and returns a -1 (error) if so. If no limit problem
sets the CCD ADC gain using the **CGAIN** and emits an 0
(no error) and the the terminating character.

DARK   Gets the exposure time in deciseconds, checks and makes
sure there is no zero-length exposure returning a -1 (error).
If it is a valid exposure time a dark exposure is taken and a
terminating character is emitted.

DEFAULT   Sets the CCD to the default readout parameters initially set
by Advanced Technologies. The CCD will subsequently be
readout as a 1x1 binned, 512x512 image.

EXPOSE   Gets the exposure time in deciseconds, checks to see if the
exposure is < 2 deciseconds returning a -1 (error) if true.
If it is a valid exposure time the shutters are closed (with no
status check), the CCD is cleared, and the shutters are
opened. After 100 msec the shutters are checked and error
codes are returned if they are not both opened (B2 flags
error, B0, B1 give shutter status). If both shutters are
opened, the expose time minus 100 msec is waited and the
shutters are subsequently closed. The CCD is not read.

189

FOCUS
Focus is not presently implemented and just emits the terminating character. The read rate of the MIP is so fast, that it has not been necessary to implement. Also, the user can use the aquisition table to continuously run an entry if focus is really desired.

OBS
The basic sequence for an 'observe' is:

    (o) get exposure time in deciseconds

    (i) check exposure limit > 100 msec...

    (ii) close the the shutters and clear the CCD...

    (iii) open the front shutter and check...

    (iv) read the intensifier photodiode and check
          for AGC saturation...

    (v) open the back shutter and check...

    (vi) wait for x deciseconds...

    (vii) close shutters, check and read CCD...

The return codes for the errors are:

    (i) **-1**   (iii) **-2**   (iv) **-4**   (v) **-8**   (vii) **-16**

'OBS'erve will execute according to the error reporting level set as discussed in 7.3.3. At termination of OBS, a termination character is emitted.

SET-ERR
SET-ERR accepts a new error reporting level, checks for limit errors (0-->2) and reports a **-1** if there was an error. If not, the new error level is stored in ERRLVL and a **0** is emitted alc... with the terminating character.

HWED
Commands the DSP to create a horizontal wedge pattern and then emits a terminating character. This shade is generated by the DSP chip to increase the pixel intensity every 64 pixels starting at an intensity of 0. This gives a pattern with every pixel value from 0 to 4095 (12 bits). This command is useful to quickly test the camera operation and the digital interface. HWED emits a terminating character.

CCD_CLR   Clears the CCD, and emits a terminating character.

CCD_READ   Reads the CCD, and emits a terminating character.

SET_FILPOS   Gets a number from the host, checks for limits and returns a -6 if there was an input error. Gets the last command from MIPCTL, masks in the new filter wheel target (B0-B2) and outputs this to the Interface board. The Interface board is then read every 20 msec. The filter wheel position is masked out and checked with the target position. If they are not equal, this process is repeated until the filter wheel has reached the target, or 1 second has elapsed. If the filter wheel reaches the target, the new present position read is returned along with the terminating character. If not, the filterwheel position is negated and returned along with the terminating character.

SET_INTGAIN   Gets a number from the host, checks for limits and returns a -6 if there was an input error. The command is then shifted into the output byte (B3-B4) and outputted it to the Interface card. After 1 msec, the interface card is read (**IG**), compared with the output cmd, returns a 0 if they're equal or a **-IG** if they're not. Finally, SET_INTGAIN emits a terminating character.

INT_ON   Turns on the Intensifier Power control bit in the output byte (B5), outputs it to the Interface card, and returns the terminating character.

INT_OFF   Turns off the Intensifier Power control bit in the output byte (B5), outputs it to the Interface card, and returns the terminating character.

RD_INTGAIN   Reads the status byte from the Interface card, masks out the intensifier gain bits (B4-B5), returns this value, and emits a

terminating character.

RD_FILPOS   Reads the status byte from the Interface card, masks out the
filter wheel position bits, returns this value, and emits a
terminating character.

SET_MIPBYTE Gets a number from the host, outputs this to the Interface
card and emits a terminating character. Does no error
checking to make sure this number is valid.

RD_MIPBYTE   Reads the Interface status byte, returns this to the host and
emits the terminating character.

RD_POSERR   Reads the Interface status btye, masks out the position error
bit (B3), returns this to the host and emits the terminating
character.

RD_TEMPS   Sets up the 68HC11 ADC to read PE0 - PE3 (the
temperature channels), outputs these four temperature
values to the host, and emits the terminating character.
Each channel is read separately four times and averaged.
The output is read as: **CCD TEC INT FILT**

RD_INTBRT   Sets up the 68HC11 ADC to read PE4 four times (the
intensifier brightness as measured by the photodiode),
takes an average and outputs this to the host, and emits
the terminating character.

7.3.5 MIP Command Loop Words

MIP_CMD   MIP_CMD is set up as a command lookup table that
executes a command from a list based on an offset number
on the stack.; i.e. 4 MIP_CMD would execute the fifth
command word defined in the :CASE statement.

MIP
MIP is the actual command loop that is executed upon power-up or reset. This loop just waits for a command, check the limits for a valid command and then converts this command into an offset for the command list MIP_CMD. Once the command is executed, MIP checks to see if the offset was 0 (an "exit loop" command), and if so exits to the FORTH environment. If not, there is space to execute the system cop utility. This is not presently implemented.

7.3.6 Autostart handling Words

MIPINIT
MIPINIT is the initialization routine for the system. This is the first word to get executed upon power-up. MIPINIT retrieves the FORTH dictionary from the EEPROM and moves it into the system RAM. The hardware is initialized using the FORTH word INIT which loads up the the DSP code into its RAM, and then initializes the DSP. Once this is completed, the command loop MIP is initiated.

MSTART!
MSTART! stores the starting address of the initialization routine MIPINIT in the autostart location of the 68HC11. This is the first address that is looked at upon power-up and is used to execute MIPINIT. This function should always be last in the dictionary and executed after a STORE to preserve the new dictionary structure.

7.4 System Cop explanation

The system cop is a feature of the 68HC11 that allows the processor to reset itself if the program hangs up. This is done by having your program constantly writing a known pattern to a reserved location. If your program hangs, this location is no longer modified and the 68HC11 recognizes that the program has hung, initiating a COLD boot. This feature needs to be implemented using the WIPE utility which is located on the IBM PC. Unfortunately, to enable and disable the system cop, you need to write to it's registers within the first few system ticks (see WIPE documentation). This is all done via the serial communications, and is not that easy.

In view of this, the system cop is not presently implemented. If it seems that it is desirable, the MIP loop has provision for it. In addition, any other timing functions would have to be modified (such as WAIT's, and DSP_WAIT). For more information, read the 68HC11 reference manual, the MaxFORTH manual, the WIPE documentation, and if still confused, call New Micro!

7.5 MIP Command Format Summary from Host's point of view

(C Programming format)

Command:	Value:	InParam	OutParam	Termination
COLD	'1'	None	%s	None
CCD_BIAS	'2'	None	None	'#'
CCD_READ	'3'	None	None	'#'
CCD_CLR	'4'	None	None	'#'
1BIN	'5'	None	None	'#'
2BIN	'6'	None	None	'#'
DEFAULT	'7'	None	%s	'#'

Command:	Value:	InParam	OutParam	Termination
CCD_GAIN	'8'	%d\r	%d	'#'
DARK	'9'	%d\r	%d	'#'
OBS	':'	%d\r	%d	'#'
EXPOSE	';'	%d\r	%d	'#'
FOCUS	'<'	None	None	'#'
HWED	'='	None	None	'#'
SET-ERR	'>'	%d\r	%d	'#'
DWED	'?'	None	None	'#'
OPEN_SH1	'@'	None	%d	'#'
OPEN_SH2	'A'	None	%d	'#'
CLOSE_SH1	'B'	None	%d	'#'
CLOSE_SH2	'C'	None	%d	'#'
OPEN_SHTS	'D'	None	%d	'#'

194

CLOSE_SHTS	'E'	None	%d	'#'
SET_FILPOS	'F'	%d\r	%d	'#'
SET_INTGAIN	'G'	%d\r	%d	'#'
SET_MIPBYTE	'H'	%d\r	None	'#'
INT_ON	'I'	None	None	'#'
INT_OFF	'J'	None	None	'#'
RD_SHTRS	'K'	None	%d	'#'
RD_FILPOS	'L'	None	%d	'#'
RD_MIPBYTE	'M'	None	%d	'#'
RD_INTGAIN	'N'	None	%d	'#'
RD_POSERR	'O'	None	%d	'#'
RD_TEMPS	'P'	None	%d_%d_%d_%d_	'#'
RD_INTBRT	'Q'	None	%d	'#'

7.6 Advanced Technologies CCD Format parameters

The CCD read format parameters are located in the KEOcmd7 code:

VARIABLE	SLENGTH	; Serial length
VARIABLE	PLENGTH	; Parallel length
VARIABLE	SBINNUM	; Serial binning factor
VARIABLE	PBINNUM	; Parallel binning factor
VARIABLE	SPRESCAN	; Number of serial pixels to shift before
		; start of read
VARIABLE	PPRESCAN	; Number of parallel shifts to do before
		; start of read
VARIABLE	SPOSTSCAN	; Number of serial shifts to do after
		; end of read
VARIABLE	PPOSTSCAN	; Number of parallel shifts to do after
		; end of read
VARIABLE	SREADLEN	; Number of serial pixels to read
VARIABLE	PREADLEN	; Number of parallel rows to read

The CCD format parameters are initialized as:

```
DECIMAL
516    SLENGTH    !
516    PLENGTH    !
1      SBINNUM    !
1      PBINNUM    !
20     SPRESCAN   !              ; This is a constant due to physical
                                 ; extension in the serial register
0      SPOSTSCAN  !
0      PPRESCAN   !
0      PPOSTSCAN  !
516    SREADLEN   !
516    PREADLEN   !
```

The **INIT-FORMAT** command takes the values presently stored in the above variables and stores them in the DSP chip. To change the readout of the CCD, any of the above can be changed, and then **INIT-FORMAT** executed.

7.7 Recovering from a ROM Crash

If you think that the ROM on the Advanced Technologies Controller board has been corrupted, here is a recovery procedure:

If you connect the camera to a terminal and then reset the 68HC11, and do not get a response (i.e. the startup text does not appear), then you should first suspect the ROM. Of course, there could be many other errors that a corrupted ROM might produce, but this procedure addresses a fatal error.

Upon resetting the 68HC11, it tries to load the code in the ROM (U15) into it's RAM (U8). If there is a problem, it will never succeed and hence never get into the interactive mode. The MIP autostart routine boots the FORTH chip, restores the ROM, initializes the DSP chip, and then starts the MIP command loop, which leaves a "#" on the terminal and waits for the next valid command as defined by the MIP code. A "0" gets you back to the FORTH kernel, and a "1" gives you a cold boot. If these don't work, chances are the ROM has been corrupted.

The first step to rebuilding the ROM is to remove U15 (the ROM) and gently bend pin 20 (chip select) up. Using small clip leads connect this pin to +5V through a 10K resistor. Now power up the board. You should see

MaxForth 3.3

This tells you that the 68HC11 is working from it's internal ROM which has the FORTH kernel in it. If you don't get this, there must be another problem with the board, or perhaps even in your communications set up. (Is your terminal set correctly? 9800B, 8Bits, 1Stop, No Parity).

If you do get this identifier, you are now ready to download the KEO Forth code. Start with an ASCII send of KEOASM7. As each   es, it will prompt you to send the next file to the 68HC11. After you have fi...... '.   nding all the ASCII files, the program is in the controller's RAM (U8).

The next step is to download the DSP hex code. Type the following command to the 68HC11:

DNLD

The 68HC11 is now waiting for an ASCII file containing the HEX code for the DSP. Send the file: KEO7.DSP. A stream of HEX numbers should appear on the terminal. This is terminated by a $ upon which the 68HC11 should output:

DOWNLOAD COMPLETED

Hit a couple of <cr> to make sure you still get the "OK" prompt.
Type:
INIT

and watch the DSP chip initialize. If it completes this, your board is now working again.

At this point though, it is necessary to store this recreated code in RAM, back into ROM. Presently the ROM is disabled because the chip select pin is pulled high through the 10K resistor. With the board still powered on, very carefully disconnect the pullup resistor and connect the pin back into it's socket. Don't try bend it, just clip on to it. Now

197

the ROM chip select is reconnected and you can store the RAM dictionary into the ROM.
Type the command:

STORE
STORING DICTIONARY

After a few minutes, the 68HC11 will output:

STORE COMPLETED

At this point, it is necessary to store the autostart routine:

MSTART!

If all has gone well, you have now successfully recovered the 68HC11 ROM. To test, power down the controller board (or just do a reset or COLD) and make sure that all the start prompts appear and that the processor comes up in the MIP command mode as usual.

<u>Order for sending ASCII text files to the 68HC11</u>

- KEOasm7
- KEOrom7
- KEOdsp7
- KEOadc8
- KEOtim7
- KEOcmd7
- MIP10.FOR
- Download: KEO8.DSP

Chapter 8 FORTH Code Listings

8.1 MIP10.FOR *KEO Consultants* Code for Camera Controller

```
( MIP10.FOR                                     KEO CONSULTANTS )
(                              LAST MODIFIED BY:   CAL07/23/93  )
( 07/23 mod:         Added better resolution to Set_filpos          )
( 09/21 mod:         Added error level system                       )
( 09/21 mod:         Changed all error codes to return a            )
(                    negative number.                               )
( 09/21 mod:         SET_FILPOS poles for correct position          )
(                    instead of just waiting 1 second               )


( New software development for MIP camera.  Interface   )
( to Master through serial port, using SYSTEM COP, and  )
( no echoing of characters.                             )
(                                                       )
( COMMENTS: Presently, the System Cop is not imple-     )
(        mented.  All functions do range checking.      )
(        Another function could be added to create      )
(        a timeout error on GETNUM to decrease the      )
(        chance of a communications lock-up.  A         )
(        new version has been written that has a        )
(        two letter command starting with the           )
(        character "C", and has been tested and         )
(        works, but I have not implemented it as        )
(        as I don't think it is presently necessary     )


( WARNING:  Do not use TABs in any FORTH file!     )
FORGET IN_BUFF

( Create an Input Buffer for incoming parameters        )

DECIMAL

CREATE IN_BUFF 5 ALLOT
IN_BUFF 5 ERASE
5 IN_BUFF C!

( Create the word to receive ASCII numbers   GETNUM   )

: GETNUM IN_BUFF 1+ 4 ERASE         ( Erase input buffer    )
    4 0 DO KEY DUP                  ( Get input character   )
    13 = IF DROP LEAVE              ( Check to see if <CR> )
    THEN IN_BUFF I + 1+ C! LOOP     ( Store the character   )
```

```
0 1 IN_BUFF CONVERT DROP DROP ;       ( Convert to num      )


HEX
B400 CONSTANT IOCTL              ( I/O CONTROL REGISTER LOCATION )
DECIMAL


VARIABLE MIPSTS                  ( LAST READ MIP STATUS )
VARIABLE MIPCTL                  ( LAST WRITTEN MIP COMMAND )


VARIABLE ERRLVL                  ( ERROR LEVEL )
VARIABLE ERRSTS                  ( ERROR STATUS WORD )

( MIP Utilities for KEO Software Control      )

: TERM_EMIT 35 EMIT ;                              ( Emit "#" to terminal )

:CASE CGAIN LO-GAIN MID-GAIN HI-GAIN ;    ( Select CCD Gain )

: CTL!     ( WRITES A BYTE TO THE IO CONTROL REGITER )
   DUP MIPCTL C! IOCTL C! ;


1 CTL!     ( INITIALIZE TO IRST FILTER, INT. OFF WITH 0 GAIN )


: CTL@      ( READS A BYTE FROM THE IO CONTROL REGISTER )
   IOCTL C@ DUP MIPSTS C! ;


: ERR! ERRSTS @ OR ERRSTS ! ;        ( OR'S IN THE PRESENT ERROR BIT )

( ERRCHK: CHECKS ERROR LEVEL, AND REPORTS AND LEAVES A CONTINUE FLAG )
(      DEPENDING ON THE ERROR REPORTING LEVEL.                       )
(      0: IGNORE ALL ERRORS                                          )
(      1: REPORT ALL ERRORS, BUT DON'T ABORT OPERATION               )
(      2: REPORT ALL ERRORS, AND ABORT OPERATION                     )

: ERRCHK  ( <condition> <err#> ERRCHK <continue flag> )
   SWAP
   IF ERRLVL @ 2 =
    IF ERR! 0
    ELSE
     ERRLVL @
      IF ERR!
      ELSE DROP
      THEN 1
    THEN
   ELSE DROP 1
   THEN ;
```

HEX

```
: SHTR_STS          ( GET THE STATUS BITS OF THE SHUTTERS  AND LEAVE ON STACK )
    CTL@ C0 AND          ( READ AND MASK INPUT BYTE )
    2/ 2/ 2/ 2/ 2/ 2/ ;     ( SHIFT DATA TO B0-B1 )

: CLS               ( CLOSE BOTH SHUTTERS WITH NO STATUS CHECK )
    PORTA C@ F-5 F-6 PORTA C! ;

: OPS               ( OPEN BOTH SHUTTERS WITH NO STATUS CHECK )
    PORTA C@ T-5 T-6 PORTA C! ;
```

(KEO Dictionary for MIP control words in order of CASE)

DECIMAL

```
: SYSTEM                        ( RETURN TO FORTH USER'S SYSTEM )
    TERM_EMIT ;

: RD_SHTRS                      ( READ THE SHUTTER STATUS BITS )
    SHTR_STS                    ( GET THE STATUS BITS ON THE STACK )
    . TERM_EMIT ;               ( RETURN STATUS )

: OPEN_SH1                      ( OPEN SHUTTER #1 )
    PORTA C@ T-6 PORTA C!       ( WRITE BIT TO PORTA )
    1 DWAIT SHTR_STS 1 AND
    IF 0 . ELSE -1 . THEN TERM_EMIT ;

: OPEN_SH2                      ( OPEN SHUTTER #2 )
    PORTA C@ T-5 PORTA C!       ( WRITE BIT TO PORTA )
    1 DWAIT SHTR_STS 2 AND
    IF 0 . ELSE -2 . THEN TERM_EMIT ;

: CLOSE_SH1                     ( CLOSE SHUTTER #1 )
    PORTA C@ F-6 PORTA C!       ( WRITE BIT TO PORTA )
    1 DWAIT SHTR_STS 1 AND
    IF -1 . ELSE 0 . THEN TERM_EMIT ;

: CLOSE_SH2                     ( CLOSE SHUTTER #2 )
    PORTA C@ F-5 PORTA C!       ( WRITE BIT TO PORTA )
    1 DWAIT SHTR_STS 2 AND
    IF -2 . ELSE 0 . THEN TERM_EMIT ;
```

```
: OPEN_SHTS                        ( OPEN BOTH SHUTTERS SIMULTANEOUSLY )
    OPS                            ( WRITE BITS TO PORTA )
    1 DWAIT SHTR_STS DUP 3 =
    IF DROP 0 . ELSE -4 OR . THEN TERM_EMIT ;

: CLOSE_SHTS                       ( CLOSE BOTH SHUTTERS SIMULTANEOUSLY )
    CLS                            ( WRITE BITS TO PORTA )
    1 DWAIT SHTR_STS DUP 3 AND
    IF -4 AND . ELSE DROP 0 . THEN TERM_EMIT ;

: CCD_BIAS                         ( READ OUT A BIAS FRAME )
    CLS CLEAR CLEAR CLEAR READ TERM_EMIT ;

: 2BIN                             ( SET TO 2x2 BINNING )
    2 SBINNUM ! 2 PBINNUM ! 256 SREADLEN ! 256 PREADLEN !
    INIT-FORMAT TERM_EMIT ;

: 1BIN                             ( SET TO 1x1 BINNING )
    1 SBINNUM ! 1 PBINNUM ! 516 SREADLEN ! 516 PREADLEN !
    INIT-FORMAT TERM_EMIT ;

: CCD_GAIN                         ( SET THE CCD A/D CONVERTER GAIN )
    GETNUM DUP 2 > IF DROP -1 . ELSE CGAIN 0 .
    THEN TERM_EMIT ;

: DARK                             ( DARK EXPOSURE IN DECISECONDS )
    GETNUM DUP 1 < IF DROP -1 .
    ELSE DDARK THEN TERM_EMIT ;

: DEFAULT                          ( SET CCD TO DEFAULT PARAMETERS )
( INITIALIZE FORMAT PARAMETERS TO DEFAULTS )
    DECIMAL
    516 SLENGTH !
    516 PLENGTH !
    1   SBINNUM !
    1   PBINNUM !
    20  SPRESCAN !
    0   SPOSTSCAN !
    0   PPRESCAN !
    0   PPOSTSCAN !
    516 SREADLEN !
    516 PREADLEN !
    INIT-FORMAT                    ( SEND THESE TO THE DSP )
    TERM_EMIT ;
```

```
: EXPOSE                          ( EXPOSE CCD )
    GETNUM DUP 2 < IF DROP -1 .
    ELSE CLS CLEAR OPS 1 DWAIT
    SHTR_STS DUP 3 = IF DROP 1 - DWAIT CLS
    1 DWAIT 0 . ELSE CLS -4 OR .
    THEN THEN TERM_EMIT ;


: FOCUS                           ( FOCUS LOOP )  ( NOT YET IMPLEMENTED )
    TERM_EMIT ;


DECIMAL
: OBS              ( 'OBSERVE': OPEN SHUTTERS SEPERATELY, CHECK PHOTODIODE )
    0 ERRSTS ! GETNUM DUP 2 <       ( GET EXPOSURE TIME IN DECISECONDS )
    1 ERRCHK                        ( CHECK FOR ERROR: LIMIT CHECK    )
    IF DUP 2 <                      ( MAKE SURE <EXP> IS > 1          )
      IF DROP 2 THEN

      CLS CLEAR                     ( SHTS CLOSED? CLEAR CCD          )
      PORTA C@ T-6 PORTA C!         ( OPEN FRONT SHUTTER: SH1         )
      1 DWAIT SHTR_STS 1 AND 0=     ( WAIT AND CHECK IF OPEN          )

      2 ERRCHK                      ( CHECK FOR ERROR: OPENING SH1    )

      IF
        SET-NOSCAN SET-SINGLE       ( SET UP ADC MODE                 )
        SET-CH4 4 ADC@              ( READ ADC CHANNEL 4              )
        + + + 4 / DUP               ( TAKE AVERAGE                    )

        180 > 4 ERRCHK              ( CHECK FOR ERROR: INT. IN AGC MODE )
        IF
          PORTA C@ T-5 PORTA C!     ( IF NOT OPEN BACK SHTR: SH2      )
          1 DWAIT SHTR_STS 2 AND 0=  ( WAIT AND CHECK IF OPEN          )

          8 ERRCHK                  ( CHECK FOR ERROR: OPENING SH2    )
          IF
            SWAP 1 - DWAIT CLS READ  ( WAIT <N> DSECS, CLOSE SHTS, READ CCD )
            SHTR_STS 16 ERRCHK DROP  ( CHECK FOR ERROR: CLOSING SHUTTERS )

          ELSE DROP CLS THEN         ( DROP <AGC>: CLS ERRCHK          )
        ELSE DROP CLS THEN           ( DROP <AGC>: AGC ERRCHK          )
      ELSE CLS THEN                  ( CLOSE SHTS: SH1 ERRCHK          )
    THEN                             ( LIMIT ERRCHK                    )

    ERRLVL @ 0=                      ( IF NO ERROR REPORT...           )
    IF .                             ( OUTPUT AGC VALUE                )
```

```
        ELSE ERRSTS @ DUP 0=           ( ELSE, CHECK ERROR WORD          )
         IF DROP .                     ( NO ERROR, DROP ERROR AND RETURN AGC )
         ELSE NEGATE . DROP THEN       ( RETURN -ERROR CODE AND DROP AGC     )
        THEN TERM_EMIT ;               ( AND RETURN TERMINATING CHARACTER )


: SET-ERR                             ( SET ERROR LEVEL    )
    GETNUM DUP                        ( GET DESTINATION    )
    2 > IF DROP -1  ELSE DUP          ( ERRLVL > 2, ERROR  )
    0 < IF DROP -1  ELSE              ( ERRLVL < 0, ERROR  )
    ERRLVL ! 0 THEN THEN             ( STORE THE ERRLVL   )
    . TERM_EMIT ;                     ( RETURN WITH CODE   )

: HWED                    ( HORIZONTAL WEDGE 12 BIT TEST PATTERN )
    SHADE TERM_EMIT ;

: DWED                    ( DIAGONAL WEDGE 12 BIT TEST PATTERN )
    ( NOT YET IMPLEMENTED ) TERM_EMIT ;

: CCD_CLR                             ( CLEAR THE CCD                )
    CLEAR TERM_EMIT ;

: CCD_READ                            ( READ OUT THE CCD             )
    READ TERM_EMIT ;

HEX
: SET_FILPOS              ( SET FILTER POSITION AND WAIT FOR FILTER WHEEL )
    GETNUM DUP 5 >                    ( GET DESTINATION              )
    IF DROP -6                        ( DESTINATION > 5, ERROR       )
    ELSE DUP 1 <                      ( DESTINATION < 1, ERROR       )
    IF DROP -6
    ELSE
      MIPCTL C@ F8 AND OR             ( SET WITH PREVIOUS COMMAND        )
      CTL!                            ( STORE NEW COMMAND AND OUTPUT  )

      32 0 DO                         ( SET UP A WAIT LOOP FOR MOVE    )
        14 MWAIT                      ( WAIT 20 MILLISECONDS          )
        CTL@ 07 AND DUP               ( GET INPUT BYTE AND MASK       )
        MIPCTL C@ 07 AND =            ( COMPARE WITH FW COMMAND       )
        IF LEAVE                      ( TERMINATE THE LOOP: FW PC   ON STACK )
        ELSE I 31 =                   ( CHECK TO NUMBER OF WAITS...      )
          IF NEGATE LEAVE THEN        ( TIMEOUT ERROR: -FW POS ON STACK  )
        THEN DROP LOOP                ( END OF WAIT FOR MOVEMENT LOOP   )

    THEN                              ( LIMIT CHECKS IF-ELSE-THEN'S     )
```

```
THEN . TERM_EMIT ;                    ( RETURN                              )

: SET_INTGAIN                    ( SET INTENSIFIER GAIN AND CHECK INTENSIFIER )
    GETNUM DUP                   ( GET INTENSIFIER GAIN          )
    3 > IF DROP -6 ELSE DUP      ( GAIN > 3, ERROR               )
    0 < IF DROP -6 ELSE          ( GAIN < 0, ERROR               )
    HEX 3 AND 2* 2* 2*           ( MASK AND SHIFT INPUT          )
    MIPCTL C@ E7 AND OR          ( SET PREVIOUS COMMAND          )
    CTL!                         ( OUTPUT TO INTERFACE           )
    1 MWAIT                      ( WAIT A MILLISECOND            )
    CTL@ 30 AND DUP              ( GET INPUT BYTE AND MASK       )
    2/ 2/ 2/ 2/ SWAP             ( LEAVE SHIFTED INT GAIN ON STACK   )
    MIPCTL C@ 18 AND 2*          ( COMPARE WITH MASKED COMMAND )
    = NOT IF NEGATE THEN         ( CREATE ERROR CODE: -INTGAIN   )
    THEN THEN                    ( LIMIT CHECK IF-ELSE-THEN'S    )
    DECIMAL . TERM_EMIT ;        ( RETURN                        )

: INT_ON                         ( TURN ON THE IMAGE INTENSIFIER   )
    MIPCTL C@ T-5 CTL! TERM_EMIT ;

: INT_OFF                        ( TURN OFF THE IMAGE INTENSIFIER )
    MIPCTL C@ F-5 CTL! TERM_EMIT ;

: RD_INTGAIN                     ( READ THE INTENSIFIER GAIN     )
    CTL@ 30 AND                  ( READ AND MASK INPUT BYTE      )
    2/ 2/ 2/ 2/                  ( SHIFT DATA TO B0-B1           )
    . TERM_EMIT ;                ( RETURN GAIN                   )

: RD_FILPOS                      ( READ THE FILTER WHEEL POSITION    )
    CTL@ 7 AND                   ( READ AND MASK INPUT BYTE      )
    . TERM_EMIT ;                ( RETURN POSITION               )

: SET_MIPBYTE                    ( SET THE MIP OUTPUT BYTE       )
    GETNUM CTL! TERM_EMIT ;      ( OUTPUT BYTE AND RETURN        )

: RD_MIPBYTE                     ( READ THE MIP INPUT BYTE       )
    CTL@ DECIMAL . TERM_EMIT ;   ( READ INPUT BYTE AND RETURN    )

: RD_POSERR                      ( READ THE FW POS. ERROR BIT    )
    CTL@ MASK-3                  ( READ INPUT BYTE, MASK         )
    0> . TERM_EMIT ;             ( AND RETURN STATUS             )

DECIMAL
```

```
: RD_TEMPS DECIMAL                     ( READ THE FOUR SYSTEM
TEMPERATURES )
    SET-NOSCAN SET-SINGLE           ( SET UP ADC MODE              )
    SET-CH0 4ADC@ + + + 4 / .       ( READ CCD  TEMP              )
    SET-CH1 4ADC@ + + + 4 / .       ( READ TEC  TEMP              )
    SET-CH2 4ADC@ + + + 4 / .       ( READ INT  TEMP              )
    SET-CH3 4ADC@ + + + 4 / .       ( READ FILT TEMP             )
    TERM_EMIT ;                     ( AND RETURN VALUES          )


: RD_INTBRT DECIMAL        ( READ THE BRIGHTNESS OF INTENSIFIER ANODE )
    SET-NOSCAN SET-SINGLE SET-CH4  ( SET UP ADC MODE             )
    4ADC@ + + + 4 / . TERM_EMIT ;  ( AND RETURN AVERAGE          )


( Set up MIP Command List .... for MIP Command loop ! )


:CASE MIP_CMD
SYSTEM      ( 0 )         COLD          ( 1 )
CCD_BIAS   ( 2 )         CCD_READ      ( 3 )
CCD_CLR    ( 4 )         1BIN          ( 5 )
2BIN       ( 6 )         DEFAULT       ( 7 )
CCD_GAIN   ( 8 )         DARK          ( 9 )
OBS        ( 10 )        EXPOSE        ( 11 )
FOCUS      ( 12 )        HWED          ( 13 )
SET-ERR    ( 14 )        DWED          ( 15 )
OPEN_SH1   ( 16 )        OPEN_SH2      ( 17 )
CLOSE_SH1  ( 18 )        CLOSE_SH2     ( 19 )
OPEN_SHTS  ( 20 )        CLOSE_SHTS    ( 21 )
SET_FILPOS ( 22 )        SET_INTGAIN   ( 23 )
SET_MIPBYTE ( 24 )       INT_ON        ( 25 )
INT_OFF    ( 26 )        RD_SHTRS      ( 27 )
RD_FILPOS  ( 28 )        RD_MIPBYTE    ( 29 )
RD_INTGAIN ( 30 )        RD_POSERR     ( 31 )
RD_TEMPS   ( 32 )        RD_INTBRT     ( 33 ) ;


(    END OF MIP_CMD LIST    6/4/91  CAL     )


34 CONSTANT NUM_CMND


( MIP COMMAND LOOP USING SYSTEM COP    6/22/91 CAL      )


: MIP                              ( COMMAND LOOP FOR MIP USER    )
    ." MIP Control Software Version: 10 " CR
    ." Hit '0' to return to FORTH Kernel " CR
    TERM_EMIT                       ( EMIT # FOR ACKNOLEDGMENT     )
    BEGIN ?TERMINAL                 ( CHECK FOR TERMINAL           )
    IF KEY 48 - DUP                 ( NORMALIZE COMMAND FROM ASCII TO INT   )
```

```
DUP 0< NOT SWAP NUM_CMND < AND          ( CHECK FOR LIMITS  )
IF DUP MIP_CMD THEN              ( GET COMMAND AND EXECUTE   )
ELSE 1                          ( 0 FOR LOOP CONTROL        )
THEN                            ( TEST FOR "SYSTEM" COMMAND  )
WHILE                           ( IF, NOT CONTINUE LOOP      )
( EXECUTE SYSTEM COP UTITLIES )
REPEAT ;


( Note: To start the 68HC11 off in the MIP program from autostart )
(     the commands: INIT and  MIP  must be added to the end )
(     of RESTORE which is the present autostart routine.  CAL  )


HEX


: MIPINIT
  1 B400 C!                     ( SET MIP INTERFACE: F1, IG0, INT_OFF )
  HEX


  CR ." RETRIEVING THE DICTIONARY " CR   ( START THE RESTORE PROCESS )


  6104                          ( START ADDR. OF DICTIONARY IN EEPROM )
  200                           ( START ADDR. OF DICTIONARY IN RAM   )
  6100 @ 6102 @                 ( GET END AND START OF RAM DICT    )
  -                             ( SUBTRACT TO GET DICTIONARY LENGTH  )
  CMOVE                         ( MOVE DICTIONARY            )


  ." RETRIEVING FORTH USER AREA " CR


  6100 @                        ( GET THE RAM DICT END   ADDRESS   )
  6102 @                        ( GET THE RAM DICT START ADDRESS    )
  -                             ( NOW WE'VE GOT THE DICTIONARY LENGTH )
  6104 +                        ( ADD EEDICT OFFSET TO GET          )
                                ( ADDR OF USER AREA IN EEPROM       )
  6 84 CMOVE                    ( MOVE USER AREA            )


  CR ." RETRIEVE STATE COMPLETE " CR
  INIT MIP ;


HEX


EPROM CONSTANT MSTART


( Autostart Store routine -- stores the starting address of MIPINIT        )
( into the auto-start address of the EEPROM.  CAUTION ************          )
( This must come last in your dictionary, and make sure you STORE          )
( any changes to the dictionary first, before doing this and then          )
```

207

(rebooting the computer -- I learned the hard way!!!   )

```
: MSTART!
  EEUNPROT
  A44A DUP MSTART EE-!
  [ ' MIPINIT ] LITERAL CFA
  DICT-START -
  EEDICT-START +
  DUP MSTART 2+ EE-!
  EEPROT
  CR ." AUTOSTART SEQUENCE STORED " CR ;
```

End of MIP10.FOR

8.2   KEOasm7   Advanced Technologies Code

```
FORGET CODE-SUB
HEX
100 TIB !
100 TIB 2+ !
200 DP !


( KEO CAMERA CODE                                    )
(                                                    )
( FORTH CODE FOR CONTROL OF THE KEO CAMERA.          )
( THIS SOFTWARE IS DIVIDED INTO                      )


( KEOASM.FOR        FORTH ASSEMBLER CODE FOR THE KEO CAMERA SYSTEM )
( WRITTEN IN FORTH AND SUPPLIED BY NEW MICROS INC.                 )

: CODE-SUB [COMPILE] CODE-SUB [COMPILE] ASSEMBLER ; IMMEDIATE
ASSEMBLER DEFINITIONS
VARIABLE MODE
: #   00 MODE ! ;
: DIR 10 MODE ! ;
: ,X  20 MODE ! ;
: ,Y 120 MODE ! ;
: EXT 30 MODE ! ;
: ?# MODE @ 0= ;
: ?DIR MODE @ 10 = ;
: ?,X MODE @  20 = ;
: ?,Y MODE @ 120 = ;
: ?EXT MODE @ 30 = ;
: MODE-LSB MODE @ FF AND ;
: ERROR ' ID. CFA 4A + EXECUTE ;
: RANGE-C, DUP FF00 AND IF HERE 1+ SWAP - DUP ABS FF00 AND
                 IF 3 ERROR THEN THEN C, ;
: CPU <BUILDS C, DOES> C@ C, EXT ; ( SINGLE BYTE OP-CODE )
: PG-2-CPU <BUILDS C, DOES> 18 C, C@ C, EXT ; ( 18 AND SINGLE BYTE OP-
CODE )
: HHLL/LL, ?EXT IF , ELSE C, THEN EXT ;
: 2HHLL/LL, ?EXT ?# OR IF , ELSE C, THEN EXT ;
: MODE-ADJ, ( a n --- a )
  ?EXT IF OVER FF00 AND 0= IF DIR THEN THEN MODE-LSB OR C, ;
: 18,Y ?,Y IF 18 C, THEN ;
: SOK? ?# IF 3 ERROR THEN ;
: 1112-CPU <BUILDS C, DOES> 18,Y C@ MODE-ADJ, HHLL/LL, ;
```

: 0112-CPU <BUILDS C, DOES> 18,Y SOK? C@ MODE-ADJ, HHLL/LL, ;
: 2112-CPU <BUILDS C, DOES> 18,Y C@ MODE-ADJ, 2HHLL/LL, ;
: (OP-DD-MM) 18,Y MODE-LSB IF C OR THEN C, C, C, ;
: OP-RR <BUILDS C, DOES> C@ C, RANGE-C, EXT ;
: OP-DD-MM <BUILDS C, DOES> C@ (OP-DD-MM) EXT ;
: OP-DD-MM-RR <BUILDS C, DOES> C@ (OP-DD-MM) RANGE-C, EXT ;
: CPX, ?,Y IF CD C, THEN 8C MODE-ADJ, 2HHLL/LL, ;
: LDX, ?,Y IF CD C, THEN CE MODE-ADJ, 2HHLL/LL, ;
: STX, ?,Y IF CD C, THEN SOK? CF MODE-ADJ, HHLL/LL, ;
: CPY, ?,X IF 1A ELSE 18 THEN C, CC MODE-ADJ, 2HHLL/LL, ;
: LDY, ?,X IF 1A ELSE 18 THEN C, CE MODE-ADJ, 2HHLL/LL, ;
: STY, ?,X IF 1A ELSE 18 THEN C, SOK? CF MODE-ADJ, HHLL/LL, ;
: CPD, ?,Y IF CD C, ELSE 1A C, THEN 83 MODE-ADJ, 2HHLL/LL, ;
00 CPU TEST,
01 CPU NOP,
02 CPU IDIV,
03 CPU FDIV,
04 CPU LSRD,
05 CPU ASLD, 05 CPU LSLD,
06 CPU TAP,
07 CPU TPA,
08 CPU INX,
09 CPU DEX,
0A CPU CLV,
0B CPU SEV,
0C CPU CLC,
0D CPU SEC,
0E CPU CLI,
0F CPU SEI,
10 CPU SBA,
11 CPU CBA,
12 OP-DD-MM-RR BRSET,
13 OP-DD-MM-RR BRCLR,
14 OP-DD-MM BSET,
15 OP-DD-MM BCLR,
16 CPU TAB,
17 CPU TBA,
(18 PAGE 2)
19 CPU DAA,
(1A PAGE 3)
1B CPU ABA,
(1C BSET,
(1D BCLR,
(1E BRSET,
(1F BRCLR,

```
20 OP-RR BRA,
21 OP-RR BRN,
22 OP-RR BHI,
23 OP-RR BLS,
24 OP-RR BCC,
24 OP-RR BHS,
25 OP-RR BCS,
25 OP-RR BLO,
26 OP-RR BNE,
27 OP-RR BEQ,
28 OP-RR BVC,
29 OP-RR BVS,
2A OP-RR BPL,
2B OP-RR BMI,
2C OP-RR BGE,
2D OP-RR BLT,
2E OP-RR BGT,
2F OP-RR BLE,
30 CPU TSX,
31 CPU INS,
32 CPU PULA,
33 CPU PULB,
34 CPU DES,
35 CPU TXS,
36 CPU PSHA,
37 CPU PSHB,
38 CPU PULX,
39 CPU RTS,
3A CPU ABX,
3B CPU RTI,
3C CPU PSHX,
3D CPU MUL,
3E CPU WAI,
3F CPU SWI,
40 CPU NEGA,
( 41 NC )
( 42 NC )
43 CPU COMA,
44 CPU LSRA,
( 45 NC )
46 CPU RORA,
47 CPU ASRA,
48 CPU ASLA,
49 CPU ROLA,
4A CPU DECA,
```

```
( 4B NC )
4C CPU INCA,
4D CPU TSTA,
( 4E NC )
4F CPU CLRA,
50 CPU NEGB,
( 51 NC )
( 52 NC )
53 CPU COMB,
54 CPU LSRB,
( 55 NC )
56 CPU RORB,
57 CPU ASRB,
58 CPU ASLB,
59 CPU ROLB,
5A CPU DECB,
( 5B NC )
5C CPU INCB,
5D CPU TSTB,
( 5E NC )
5F CPU CLRB,
( 60-7F )
40 1112-CPU NEG,
43 1112-CPU COM,
44 1112-CPU LSR,
46 1112-CPU ROR,
47 1112-CPU ASR,
48 1112-CPU ASL,
49 1112-CPU ROL,
4A 1112-CPU DEC,
4C 1112-CPU INC,
4D 1112-CPU TST,
4E 1112-CPU JMP,
4F 1112-CPU CLR,
( 80-BF )
80 1112-CPU SUBA,
81 1112-CPU CMPA,
82 1112-CPU SBCA,
83 2112-CPU SUBD,
84 1112-CPU ANDA,
85 1112-CPU BITA,
86 1112-CPU LDAA,
87 0112-CPU STAA,
88 1112-CPU EORA,
89 1112-CPU ADCA,
```

```
8A 1112-CPU ORAA,
8B 1112-CPU ADDA,
( 8C CPX,
8D 0112-CPU JSR,
8E 2112-CPU LDS,
8F 0112-CPU STS,
8F CPU XGDX,
( C0-FF )
C0 1112-CPU SUBB,
C1 1112-CPU CMPB,
C2 1112-CPU SBCB,
C3 2112-CPU ADDD,
C4 1112-CPU ANDB,
C5 1112-CPU BITB,
C6 1112-CPU LDAB,
C7 0112-CPU STAB,
C8 1112-CPU EORB,
C9 1112-CPU ADCB,
CA 1112-CPU ORAB,
CB 1112-CPU ADDB,
CC 2112-CPU LDD,
CD 0112-CPU STD,
( CE LDX, )
( CF STX, )
CF CPU STOP,
08 PG-2-CPU INY,
09 PG-2-CPU DEY,
30 PG-2-CPU TSY,
35 PG-2-CPU TYS,
38 PG-2-CPU PULY,
3A PG-2-CPU ABY,
3C PG-2-CPU PSHY,
8F PG-2-CPU XGDY,
8D OP-RR BSR,
: TOP ,Y   0 ;   ( ADDRESS THE BOTTOM OF THE STACK *)
: SEC ,Y   2 ;      ( ADDRESS SECOND ITEM ON STACK  *)
: ?EXEC STATE @ IF 12 ERROR THEN ;
: ?PAIRS - IF 13 ERROR THEN ;
: BEGIN,  HERE 1 ;
: UNTIL,  ?EXEC >R 1 ?PAIRS R> C, HERE  1+ - C, ;
: AGAIN,  21 UNTIL, ;
: IF,    C, HERE 0 C, 2 ;
: THEN,   ?EXEC 2 ?PAIRS  HERE OVER 1+ - SWAP C! ;
: ELSE,   2 ?PAIRS HERE 1+ 0 BRA,
      SWAP HERE OVER 1+ - SWAP  C!  2 ;
```

```
: .NOT. 1 XOR ;                    ( REVERSE ASSEMBLY TEST )
20 CONSTANT .FL.
21 CONSTANT .TR.
22 CONSTANT .LS.
23 CONSTANT .HI.
24 CONSTANT .CS.
24 CONSTANT .LO.
25 CONSTANT .CC.
25 CONSTANT .HS.
26 CONSTANT .EQ.
27 CONSTANT .NE.
28 CONSTANT .VS.
29 CONSTANT .VC.
2A CONSTANT .--.
2B CONSTANT .++.
2C CONSTANT .LT.
2D CONSTANT .GE.
2E CONSTANT .LE.
2F CONSTANT .GT.
'@  CFA FE43 FE22 - + CONSTANT PUSHD
'@  CFA FE47 FE22 - + CONSTANT NEXTSD
'@  CFA FE47 FE22 - + CONSTANT PUT
'@  CFA FE4A FE22 - + CONSTANT NEXT
'@  CFA FE4C FE22 - + CONSTANT NEXT3
'@  CFA FE50 FE22 - + CONSTANT NEXT1
'@  CFA FE52 FE22 - + CONSTANT NEXT2
' 1+ CFA FC97 FC7C - + CONSTANT POP
' 1+ CFA FC93 FC7C - + CONSTANT POPTWO
0 CONSTANT W
2 CONSTANT IP
4 CONSTANT UP
FORTH DEFINITIONS
ASSEMBLER DEFINITIONS
: POPD, TOP LDD, INY, INY, ;
: PUTD, DEY, DEY, TOP STD, ;
FORTH DEFINITIONS

( PLEASE DOWNLOAD KEOROM FILE )
```

8.3 KEOrom7   Advanced Technologies Code

```
( KEOROM.FOR                                                    )
( FORTH WORDS FOR DEALING WITH THE EEPROM ON THE KEO CONTROL    )
(                                                               )
(   REVISION 1 : 3.23.91                                        )
(   BROKE THIS CODE OUT OF KEO6.FOR                             )

FORGET SP


: SP   20 EMIT ;


( THESE WORDS DEAL WITH THE OFF CHIP EEPROM WHERE WE STORE THE DEFAULT )
( SYSTEM DICTIONARY                                             )

HEX


( EEPROT AND EEUNPROT ARE USED TO WRITE PROTECT THE EEPROM      )
( THERE ARE SOME FUNNY NUMBERS IN HERE WHICH ARE HARD CODED     )
( AND ASSUME THAT THE EEPROM STARTS AT $6000, WHICH IT DOES     )
( RIGHT AT THE MOMENT.                                          )
( THE ADDRESSES WE WANT TO WRITE TO, FROM THE CHIPS POINT       )
( OF VIEW, ARE   $5555 AND $2AAA. THESE ADDRESSES ARE NOT       )
( AVAILABLE FOR OUR EEPROM, BUT THE BOTTOM 14 BITS MUST BE      )
( REPRESENTING THESE ADDRESSES, SO WE WILL SET THE TOP BIT      )
( WHICH THE EEPROM DOES NOT EVEN SEE AND WRITE TO IT AT         )
( $D555 AND $AAAA   .                                           )

CODE-SUB EEPROT
    CE C,   D5 C,   55 C,          ( LDX   #SD555  1ST EPROM ADDRESS   )
    86 C,   AA C,                  ( LDAA  #SAA    1ST DATA PATTERN    )
    A7 C,   00 C,                  ( STAA  0,X     SEND AN AA TO $5555 )
    CE C,   AA C,   AA C,          ( LDX   #SAAAA  2ND EPROM ADDRESS   )
    86 C,   55 C,                  ( LDAA  #S55    2ND DATA PATTERN    )
    A7 C,   00 C,                  ( STAA  0,X     SEND A 55 TO $2AAA  )
    CE C,   D5 C,   55 C,          ( LDX   #SD555  3RD EPROM ADDRESS   )
    86 C,   A0 C,                  ( LDAA  #SA0    3RD DATA PATTERN    )
    A7 C,   00 C,                  ( STAA  0,X     SEND AN A0 TO $5555 )
    39 C,                          ( RTS           RETURN             )
END-CODE


CODE-SUB EEUNPROT
    CE C,   D5 C,   55 C,          ( LDX   #SD555  1ST EPROM ADDRESS   )
    86 C,   AA C,                  ( LDAA  #SAA    1ST DATA PATTERN    )
    A7 C,   00 C,                  ( STAA  0,X     SEND AN AA TO $D555 )
    CE C,   AA C,   AA C,          ( LDX   #SAAAA  2ND EPROM ADDRESS   )
```

```
86 C,   55 C,                (LDAA  #S55    2ND DATA PATTERN   )
A7 C,   00 C,                (STAA  0,X     SEND A 55 TO $AAAA   )
CE C,  D5 C,  55 C,          (LDX   #SD555  3RD EPROM ADDRESS   )
86 C,   80 C,                (LDAA  #S80    3RD DATA PATTERN    )
A7 C,   00 C,                (STAA  0,X     SEND AN 80 TO $D555   )
CE C,  D5 C,  55 C,          (LDX   #SD555  4TH EPROM ADDRESS   )
86 C,   AA C,                (LDAA  #SAA    4TH DATA PATTERN    )
A7 C,   00 C,                (STAA  0,X     SEND AN AA TO $D555   )
CE C,   AA C,   AA C,        (LDX   #SAAAA  5TH EPROM ADDRESS   )
86 C,   55 C,                (LDAA  #S55    5TH DATA PATTERN    )
A7 C,   00 C,                (STAA  0,X     SEND A 55 TO $AAAA    )
CE C,  D5 C,  55 C,          (LDX   #SD555  6TH EPROM ADDRESS   )
86 C,   20 C,                (LDAA  #S20    6TH DATA PATTERN    )
A7 C,   00 C,                (STAA  0,X     SEND A 20 TO $D555    )
39 C,                        (RTS           RETURN              )
END-CODE
```

(EPROM REPRESENTS WHERE IN EEPROM WE WANT TO START STORING THE DICTIONARY)
(WE WANT TO RESERVE SOME SPACE FOR AUTOSTART ROUTINES ETC SO WE START AT)
(256 BYTES ABOVE THE ACTUAL START OF THE EEPROM OR $6100   )

6000 CONSTANT EPROM
6104 CONSTANT EEDICT-START
200   CONSTANT DICT-START

: EE-!   2DUP   ! BEGIN 2DUP   @ = UNTIL DROP DROP ;
: EE-C!   2DUP C! BEGIN 2DUP C@ = UNTIL DROP DROP ;

: EETEST   AFFF EPROM DO AA I DUP . CR EE-C! LOOP ;

(STORE THE CURRENT STATE OF THE FORTH MACHINE   )
(AND THE USER ADDED DICTIONARY TO EXTERNAL EPROM   )

VARIABLE PROMLOC

: STORE
  EEUNPROT
  EPROM 100 + PROMLOC !
  (WE WILL STORE THE DICTIONARY END ADDRESS AT S6100)
  HERE PROMLOC @ EE-!

  (INCREMENT THE PROMLOC POINTER BY 2)
  PROMLOC @ 2+ PROMLOC !
```
216
```

```
DICT-START PROMLOC @ EE-!        ( WE WILL STORE THE DICTIONARY )
                                 ( START ADDRESS AT $6102 )


PROMLOC @ 2+ PROMLOC !           ( POINT TO THE START OF )
                                 ( THE EEPROM DICTIONARY )


." STORING DICTIONARY " CR       ( MOVE THE DICTIONARY TO EEPROM )
HERE DICT-START DO               ( ... ONE BYTE AT A TIME    )
I C@ PROMLOC @ EE-C!
PROMLOC @ 1+ PROMLOC !
LOOP

." STORING FORTH USER AREA " CR  ( WE WILL STORE THE FORTH USER )
84 6 DO                          ( AREA DIRECTLY ABOVE THE DICT )
I C@ PROMLOC @ EE-C!
PROMLOC @ 1+ PROMLOC !
LOOP

EEPROT
CR ." STORE COMPLETE " CR
." ENDED AT " PROMLOC @ U. CR
EEPROT ;



HEX

( REPLACE THE CURRENT STATE OF THE FORTH MACHINE   )
( USER ADDED DICTIONARY FROM EXTERNAL EPROM         )
  ( RAM DICT END    ADDRESS AT $6100      )
  ( RAM DICT START ADDRESS AT $6102       )
  ( EE-DICT STARTS AT ADDRESS $6104       )

: RESTORE
  HEX
  CR ." RETRIEVING THE DICTIONARY " CR

  6104                 ( START ADDR. OF DICTONARY IN EEPROM  )
  200                  ( START ADDR. OF DICTONARY IN RAM     )
  6100 @ 6102 @        ( GET END AND START OF RAM DICT       )
  -                    ( SUBTRACT TO GET DICTIONARY LENGTH   )
  CMOVE                ( MOVE DICTIONARY                     )

  ." RETRIEVING FORTH USER AREA " CR
```

```
6100 @              ( GET THE RAM DICT END   ADDRESS          )
6102 @              ( GET THE RAM DICT START ADDRESS          )
-                   ( NOW WE'VE GOT THE DICT LENGTH           )
6104 +              ( ADD EEDICT OFFSET TO GET                )
                    ( ADDR OF USER AREA IN EEPROM             )
6 84 CMOVE          ( MOVE USER AREA                          )

CR ." RETRIEVE STATE COMPLETE " CR ;

HEX


: DICT-CHK              ( END START --                          )
  CR
  DO I DUP C@
  SWAP EEDICT-START + DICT-START - C@ =
  IF
  ELSE
  I DUP . SP C@ . SP                              ( RAM ADDR, VAL )
  I EEDICT-START + DICT-START - DUP . SP C@ . SP  ( ROM ADDR, VAL )
  ." ?????? " CR THEN
  ( ERROR MSG )
  100 0 DO LOOP
  LOOP ;

( PLEASE DOWNLOAD THE FILE KEODSP )
```

8.4 KEOdsp7   Advanced Technologies Code

```
(                              KEODSP.FOR                              )
( FORTH WORDS FOR DEALING WITH THE DSP ON THE KEO CAMERA CONTROLLER )
(        REVISION 1 : 3.23.91                                       )
(              BROKE THIS CODE OUT OF KEO6.FOR                       )

FORGET FALSE

0 CONSTANT FALSE
1 CONSTANT TRUE


B000 CONSTANT PORTA
B002 CONSTANT PIOC
B004 CONSTANT PORTB
B003 CONSTANT PORTC
B005 CONSTANT PORTCL
B007 CONSTANT DDRC
B008 CONSTANT PORTD
B009 CONSTANT DDRD
B00A CONSTANT PORTE
B00B CONSTANT CFORC
B00C CONSTANT OC1M
B00D CONSTANT OC1D
B00E CONSTANT TCNT
B010 CONSTANT TIC1
B012 CONSTANT TIC2
B014 CONSTANT TIC3
B016 CONSTANT TOC1
B018 CONSTANT TOC2
B01A CONSTANT TOC3
B01C CONSTANT TOC4
B01E CONSTANT TOC5
B020 CONSTANT TCTL1
B021 CONSTANT TCTL2
B022 CONSTANT TMSK1
B023 CONSTANT TFLG1
B024 CONSTANT TMSK2
B025 CONSTANT TFLG2
B026 CONSTANT PACTL
B027 CONSTANT PACNT
```

```
: SET-C-IN   00 DDRC C! ;
: SET-C-OUT  FF DDRC C! ;


( BIT MANIPULATIONS )

( MASK-X LEAVES TRUE IF BIT X SET ELSE FALSE )
: MASK-7   0080 AND ;
: MASK-6   0040 AND ;
: MASK-5   0020 AND ;
: MASK-4   0010 AND ;
: MASK-3   0008 AND ;
: MASK-2   0004 AND ;
: MASK-1   0002 AND ;
: MASK-0   0001 AND ;


( CH-X CHANGES ONLY BIT X OF NUMBER LEAVING RESULT ON STACK )
: CH-7   FF7F XOR NOT ;
: CH-6   FFBF XOR NOT ;
: CH-5   FFDF XOR NOT ;
: CH-4   FFEF XOR NOT ;
: CH-3   FFF7 XOR NOT ;
: CH-2   FFFB XOR NOT ;
: CH-1   FFFD XOR NOT ;
: CH-0   FFFE XOR NOT ;


( T-X MAKES CERTAIN BIT X IS TRUE REGARDLESS OF CURRENT STATE )
: T-7   DUP MASK-7 0= IF CH-7 THEN ;
: T-6   DUP MASK-6 0= IF CH-6 THEN ;
: T-5   DUP MASK-5 0= IF CH-5 THEN ;
: T-4   DUP MASK-4 0= IF CH-4 THEN ;
: T-3   DUP MASK-3 0= IF CH-3 THEN ;
: T-2   DUP MASK-2 0= IF CH-2 THEN ;
: T-1   DUP MASK-1 0= IF CH-1 THEN ;
: T-0   DUP MASK-0 0= IF CH-0 THEN ;


( F-X MAKES CERTAIN BIT X IS FALSE REGARDLESS OF CURRENT STATE )
: F-7   DUP MASK-7 IF CH-7 THEN ;
: F-6   DUP MASK-6 IF CH-6 THEN ;
: F-5   DUP MASK-5 IF CH-5 THEN ;
: F-4   DUP MASK-4 IF CH-4 THEN ;
: F-3   DUP MASK-3 IF CH-3 THEN ;
: F-2   DUP MASK-2 IF CH-2 THEN ;
: F-1   DUP MASK-1 IF CH-1 THEN ;
: F-0   DUP MASK-0 IF CH-0 THEN ;
```

```
( ROUTINES FOR READING AND WRITING DSP HOST CONTROL AND HOST DATA )
: HB!   PORTC C! SET-C-OUT ;
: HB@   SET-C-IN PORTC C@ ;
: HCTL! PORTB C! ;
: HCTL@ PORTB C@ ;

( ROUTINES FOR SETTING THE HOST ADDRESS )
: HA0   HCTL@ F-0 F-1 F-2 HCTL! ;
: HA1   HCTL@ T-0 F-1 F-2 HCTL! ;
: HA2   HCTL@ F-0 T-1 F-2 HCTL! ;
: HA3   HCTL@ T-0 T-1 F-2 HCTL! ;
: HA5   HCTL@ T-0 F-1 T-2 HCTL! ;
: HA6   HCTL@ F-0 T-1 T-2 HCTL! ;
: HA7   HCTL@ T-0 T-1 T-2 HCTL! ;

( BIT MANIPULATIONS OF THE HOST CONTROL PORT )
: RD-EN      HCTL@ T-3 HCTL! ;
: WR-EN      HCTL@ F-3 HCTL! ;
: HACK       HCTL@ F-4 HCTL! ;
: INIT-HOST  00 HB! F8 HCTL! ;
: DSP-RESET INIT-HOST HCTL@ DUP F-6 HCTL! T-6 HCTL! ;
: DSP-IRQB  INIT-HOST HCTL@ DUP F-7 HCTL! T-7 HCTL! ;
. DSP-BOOT INIT-HOST HCTL@ DUP F-6 F-7 HCTL! DUP T-6 HCTL! T-7 HCTL!
;

: DR          DSP-RESET ;
: BOOT        DSP-BOOT ;
: EN-HOST   HCTL@ F-5 HCTL! ;
: DIS-HOST  HCTL@ T-5 HCTL! ;

: PULL-RESET INIT-HOST HCTL@ F-6 HCTL! ;
. PULL-IRQB  INIT-HOST HCTL@ F-7 HCTL! ;

( ROUTINES FOR READING AND WRITING THE DSP'S INTERNAL HOST REGISTER.
: ICR@      HA0 RD-EN EN-HOST HB@ DIS-HOST ;        (   - ICR )
: ICR!      HA0 WR-EN HB! EN-HOST DIS-HOST ;        ( BYTE -   )
: CVR@      HA1 RD-EN EN-HOST HB@ DIS-HOST ;        (   - CVR )
: CVR!      HA1 WR-EN HB! EN-HOST DIS-HOST ;        ( BYTE -   )
: ISR@      HA2 RD-EN EN-HOST HB@ DIS-HOST ;        (   - ISR )
: ISR!      HA2 WR-EN HB! EN-HOST DIS-HOST ;        ( BYTE -   )
: RXH@      HA5 RD-EN EN-HOST HB@ DIS-HOST ;        (   - RXH )
: RXM@      HA6 RD-EN EN-HOST HB@ DIS-HOST ;        (   - RXM )
: RXL@      HA7 RD-EN EN-HOST HB@ DIS-HOST ;        (   - RXL )
: TXH!      HA5 WR-EN HB! EN-HOST DIS-HOST ;        ( BYTE -   )
: TXM!      HA6 WR-EN HB! EN-HOST DIS-HOST ;        ( BYTE -   )
```

```
: TXL!        HA7 WR-EN HB! EN-HOST DIS-HOST ;         ( BYTE -   )
: CLR-TXH-TXM   00 DUP TXH! TXM! ;

( ROUTINES FOR INTERPRETING THE HOST REGISTERS )
: RXDF?    ISR@ MASK-0 ;          ( LEAVES TRUE IF RXDF BIT SET )
: TXDE?    ISR@ MASK-1 ;          ( LEAVES TRUE IF TXDE BIT SET )
: HF2?     ISR@ MASK-3 ;          ( TRUE IF HF2 SET )
: HF3?     ISR@ MASK-4 ;          ( TRUE IF HF3 SET )
: HREQ?    ISR@ MASK-7 ;          ( TRUE IF DSP IS CALLING HREQ )
: T-HF0    ICR@ T-3 ICR! ;        ( SETS HF0 TO TRUE )
: F-HF0    ICR@ F-3 ICR! ;        ( SETS HF0 TO FALSE )
: T-HF1    ICR@ T-4 ICR! ;        ( SETS HF1 TO TRUE )
: F-HF1    ICR@ F-4 ICR! ;        ( SETS HF1 TO FALSE )

: DSP-BUSY? HF2? ;
: DSP-WAIT   BEGIN DSP-BUSY? 0= UNTIL ;

: HV!                             ( VECTOR# - )
    DUP 1F < IF T-7 CVR!
        ELSE ." VECTOR TOO LARGE "
        THEN   ;


( ROUTINES FOR GETTING FILES FROM THE PC TO THE DSP )
: XLATE                      ( ASCII-VALUE -- HEX-VALUE )
    DUP DUP
    2F > SWAP 40 < AND
      IF 30 -
      ELSE DUP DUP 40 > SWAP 47 < AND
        IF 37 -
        ELSE CR .." ILLEGAL CHARACTER RECIEVED " CR
        THEN
      THEN ;




VARIABLE END-ADDR
VARIABLE BOT-BUF
VARIABLE TOP-BUF
HERE END-ADDR !
2 ALLOT
HERE BOT-BUF !
600 ALLOT
HERE TOP-BUF !
TOP-BUF @ BOT-BUF @ - CONSTANT BUF-LENGTH
: DNLD
```

```
        TOP-BUF @ BOT-BUF @
        DO KEY DUP 24 =                    ( IF CHAR = "S"                        )
          IF EMIT I 1- END-ADDR ! LEAVE ( STORE COUNTER AS END ADDR LEAVE )
          ELSE DUP 20 =                    ( IF ITS A SPACE                      )
            IF EMIT R> 1- >R               ( PRINT IT BUT DON'T COUNT IT      )
            ELSE DUP 0D =                  ( IF ITS A CR                         )
             IF EMIT 0A EMIT R> 1- >R      ( DO CR-LF DON'T COUNT IT          )
             ELSE DUP XLATE 10 * SWAP EMIT      ( ELSE XLATE MULT BY $10 )
                 ELSE CR ." DSP NOT READY   "
                 THEN
             UNTIL
             I   C@ TXH!
             I 1+ C@ TXM!
             I 2+ C@ TXL!
             3 +LOOP T-HF0 ;


: SHOW-REGS   CR
    ICR@ ." ICR =  " . CR
          KEY DUP XLATE ROT + I C! EMIT
        THEN
        THEN
       THEN
      LOOP
      CR ." DOWNLOAD COMPLETED " ;


: DSP-DUMP   BEGIN DSP-BOOT ISR@ 6 = CVR@ 12 = AND UNTIL
        END-ADDR @ BOT-BUF @ DO
            BEGIN TXDE? DUP
                IF
    ISR@ ." ISR =  " . CR
    CVR@ ." CVR =  " . CR
    RXH@ ." RXH =  " . CR
    RXM@ ." RXM =  " . CR
    RXL@ ." RXL =  " . CR ;


: SR SHOW-REGS ;


: GO    DSP-BOOT SR DSP-DUMP ;


: DSP8@                    ( READ FROM DSP - 8 BITS FROM RXL REG )
      BEGIN RXDF? UNTIL RXL@ ;
: DSP16@                   ( READ 16 BITS FROM DSP RXM AND RXL REGISTERS )
      BEGIN RXDF? UNTIL RXM@ FF ~ RXL@ + ;
```

```
: DSP8!                        ( SEND 8 BITS TO DSP TXL REGISTER )
     BEGIN TXDE? UNTIL TXL! ;
: DSP16!                       ( SEND 16 BITS TO DSP TXM AND TXL REGS )
     BEGIN TXDE? UNTIL DUP FF / TXM! TXL! ;

( PLEASE DOWNLOAD THE FILE KEOADC )
```

8.5 KEOadc8   Advanced Technologies Code

(KEOADC FORTH CODE THAT CONTROLS THE ON BOARD ANALOG TO DIGITAL CONVERTER)

FORGET ADCTL

(ANALOG TO DIGITAL CONVERTER REGISTERS)
```
B030 CONSTANT ADCTL        ( CONVERTER CONTROL REGISTER   )
B031 CONSTANT ADR1         ( RESULT REGISTER 1            )
B032 CONSTANT ADR2         ( RESULT REGISTER 2            )
B033 CONSTANT ADR3         ( RESULT REGISTER 3            )
B034 CONSTANT ADR4         ( RESULT REGISTER 4            )


: SET-SCAN   ADCTL C@ T-5 ADCTL C! ;        ( SET CONTINUOUS CONVERSIONS )
: SET-NOSCAN  ADCTL C@ F-5 ADCTL C! ;       ( SET SINGLE CONVERSION CYCLE   )

: SET-SINGLE  ADCTL C@ F-4 ADCTL C! ;       ( SETS SINGLE CHANNEL MODE      )
: SET-CH0    SET-SINGLE ADCTL C@ F-0 F-1 F-2 F-3 ADCTL C! ;
: SET-CH1    SET-SINGLE ADCTL C@ T-0 F-1 F-2 F-3 ADCTL C! ;
: SET-CH2    SET-SINGLE ADCTL C@ F-0 T-1 F-2 F-3 ADCTL C! ;
: SET-CH3    SET-SINGLE ADCTL C@ T-0 T-1 F-2 F-3 ADCTL C! ;
: SET-CH4    SET-SINGLE ADCTL C@ F-0 F-1 T-2 F-3 ADCTL C! ;
: SET-CH5    SET-SINGLE ADCTL C@ T-0 F-1 T-2 F-3 ADCTL C! ;
: SET-CH6    SET-SINGLE ADCTL C@ F-0 T-1 T-2 F-3 ADCTL C! ;
: SET-CH7    SET-SINGLE ADCTL C@ T-0 T-1 T-2 F-3 ADCTL C! ;

: SET-MULT   ADCTL C@ T-4 ADCTL C! ;   ( PUTS ADC IN MULTIPLEXED INPUT MODE )
: SET-GRP1   SET-MULT ADCTL C@ F-3 F-2 ADCTL C! ; ( USE INPUT GROUP1, CH0-3 )
: SET-GRP2   SET-MULT ADCTL C@ T-3 F-2 ADCTL C! ; ( USE INPUT GROUP2, CH4-7 )

: ADC-DONE?  ADCTL C@ MASK-7 ; ( TRUE IF A CONVERSION SEQUENCE IS COMPLETE )

( FETCH THE RESULTS OF THE CONVERSION )
: ADR1@     BEGIN ADC-DONE? UNTIL ADR1 C@ ;    ( FETCH #1 )
: ADR2@     BEGIN ADC-DONE? UNTIL ADR2 C@ ;    ( FETCH #2 )
: ADR3@     BEGIN ADC-DONE? UNTIL ADR3 C@ ;    ( FETCH #3 )
: ADR4@     BEGIN ADC-DONE? UNTIL ADR4 C@ ;    ( FETCH #4 )
: 4ADC@     BEGIN ADC-DONE? UNTIL
        ADR1 C@ ADR2 C@ ADR3 C@ ADR4 C@ ;      ( FETCH THEM ALL )

: ADTEST2   SET-SCAN 100 00 DO SET-CH0 ADR1@ . CR LOOP ;


( PLEASE DOWNLOAD KEOTIM FILE )
```

8.6 KEOtim7   Advanced Technologies Code

(KEOTIM   FORTH DEFINITIONS FOR TIMING STATES FOR KEO CAMERA)

(DEFINITIONS OF STATE CONSTANTS FOR USE IN THE KEO DSP CODE)

FORGET P1_HI

HEX

```
1     CONSTANT P1_HI    ( %0000000000000001 )
0     CONSTANT P1_LO
2     CONSTANT P2_HI    ( %0000000000000010 )
0     CONSTANT P2_LO
4     CONSTANT P3_HI    ( %0000000000000100 )
0     CONSTANT P3_LO
8     CONSTANT TG_HI         ( %0000000000001000 )
0     CONSTANT TG_LO
10    CONSTANT PIX_HI        ( %0000000000010000 )
0     CONSTANT PIX_LO
20    CONSTANT LEN_HI        ( %0000000000100000 )
0     CONSTANT LEN_LO
40    CONSTANT FEN_HI        ( %0000000001000000 )
0     CONSTANT FEN_LO
80    CONSTANT SCK_HI        ( %0000000010000000 )
0     CONSTANT SCK_LO
100   CONSTANT SW_HI         ( %0000000100000000 )
0     CONSTANT SW_LO
200   CONSTANT DCK_HI        ( %0000001000000000 )
0     CONSTANT DCK_LO
400   CONSTANT DCR_HI        ( %0000010000000000 )
0     CONSTANT DCR_LO
800   CONSTANT ADS_HI        ( %0000100000000000 )
0     CONSTANT ADS_LO
```

```
( PS VALUES ARE INTERMEDIATE STATES FOR THE PARALLEL CLOCKS )
P1_HI  P2_HI OR P3_HI OR TG_HI OR   CONSTANT PS0
P1_HI  P2_LO OR P3_HI OR TG_HI OR   CONSTANT PS1
P1_HI  P2_LO OR P3_LO OR TG_LO OR CONSTANT PS2
P1_HI  P2_HI OR P3_LO OR TG_LO OR  CONSTANT PS3
P1_LO  P2_HI OR P3_LO OR TG_LO OR CONSTANT PS4
P1_LO  P2_HI OR P3_HI OR TG_HI OR   CONSTANT PS5
P1_LO  P2_HI OR P3_HI OR TG_HI OR   CONSTANT PS6
```

P1_HI P2_HI OR P3_HI OR TG_HI OR   CONSTANT PS7
P1_HI P2_HI OR P3_HI OR TG_HI OR   CONSTANT PS8

(PAR_IDLE IS THE STATE IN WHICH TO LEAVE THE PARALLEL)
(CLOCKS WHEN THEY ARE NOT IN USE   )
P1_HI P2_HI OR P3_HI OR TG_HI OR   CONSTANT **PAR_IDLE**

(SPR VALUES ARE PRELIMINARY STATES FOR SERIAL PRESCAN PIXELS)
SCK_HI SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR0**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR1**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR2**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR3**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_HI
OR ADS_LO OR CONSTANT **SPR4**
SCK_LO SW_HI OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR5**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR6**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_HI OR DCR_LO
OR ADS_LO OR CONSTANT **SPR7**
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPR8**
SCK_LO SW_LO OR PIX_HI OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_HI OR CONSTANT **SPR9**

(SRD VALUES ARE INT. STATES FOR SERIAL READ PIXELS
SCK_HI SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD0**
SCK_LO SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD1**
SCK_LO SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD2**
SCK_LO SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD3**
SCK_LO SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_HI
OR ADS_LO OR CONSTANT **SRD4**
SCK_LO SW_HI OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD5**
SCK_LO SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD6**

SCK_LO   SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_HI OR DCR_LO
OR ADS_LO OR CONSTANT **SRD7**
SCK_LO   SW_LO OR PIX_HI OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SRD8**
SCK_LO   SW_LO OR PIX_LO OR LEN_LO OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_HI OR CONSTANT **SRD9**

(SPST VALUES ARE INT. STATES FOR THE SERIAL POSTSCAN PIXELS)
SCK_HI   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST0**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST1**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST2**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST3**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST4**
SCK_LO   SW_HI OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST5**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST6**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_HI OR DCR_LO
OR ADS_LO OR CONSTANT **SPST7**
SCK_LO   SW_LO OR PIX_HI OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SPST8**
SCK_LO   SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_HI OR CONSTANT **SPST9**

(SER_IDLE IS THE STATE TO LEAVE THE SERIAL CONTROL BITS IN WHEN NOT IN USE
SCK_LO SW_LO OR PIX_LO OR LEN_HI OR FEN_LO OR DCK_LO OR DCR_LO
OR ADS_LO OR CONSTANT **SER_IDLE**

(PSTATES ARE FINAL STATES FOR PARALLEL CLOCKING
PS0 SER_IDLE OR CONSTANT **PSTATE0**
PS1 SER_IDLE OR CONSTANT **PSTATE1**
PS2 SER_IDLE OR CONSTANT **PSTATE2**
PS3 SER_IDLE OR CONSTANT **PSTATE3**
PS4 SER_IDLE OR CONSTANT **PSTATE4**
PS5 SER_IDLE OR CONSTANT **PSTATE5**
PS6 SER_IDLE OR CONSTANT **PSTATE6**
PS7 SER_IDLE OR CONSTANT **PSTATE7**

PS8 SER_IDLE OR CONSTANT **PSTATE8**

(SPRE STATES ARE FINAL STATES FOR SERIAL PIXEL PRESCANNING
SPR0 PAR_IDLE OR CONSTANT **SPRE0**
SPR1 PAR_IDLE OR CONSTANT **SPRE1**
SPR2 PAR_IDLE OR CONSTANT **SPRE2**
SPR3 PAR_IDLE OR CONSTANT **SPRE3**
SPR4 PAR_IDLE OR CONSTANT **SPRE4**
SPR5 PAR_IDLE OR CONSTANT **SPRE5**
SPR6 PAR_IDLE OR CONSTANT **SPRE6**
SPR7 PAR_IDLE OR CONSTANT **SPRE7**
SPR8 PAR_IDLE OR CONSTANT **SPRE8**
SPR9 PAR_IDLE OR CONSTANT **SPRE9**

(SREAD STATES ARE FINAL STATES FOR SERIAL PIXEL READING
SRD0 PAR_IDLE OR CONSTANT **SREAD0**
SRD1 PAR_IDLE OR CONSTANT **SREAD1**
SRD2 PAR_IDLE OR CONSTANT **SREAD2**
SRD3 PAR_IDLE OR CONSTANT **SREAD3**
SRD4 PAR_IDLE OR CONSTANT **SREAD4**
SRD5 PAR_IDLE OR CONSTANT **SREAD5**
SRD6 PAR_IDLE OR CONSTANT **SREAD6**
SRD7 PAR_IDLE OR CONSTANT **SREAD7**
SRD8 PAR_IDLE OR CONSTANT **SREAD8**
SRD9 PAR_IDLE OR CONSTANT **SREAD9**

(SPOST STATES ARE FINAL STATES FOR SERIAL PIXEL POSTSCANNING)
SPST0 PAR_IDLE OR CONSTANT **SPOST0**
SPST1 PAR_IDLE OR CONSTANT **SPOST1**
SPST2 PAR_IDLE OR CONSTANT **SPOST2**
SPST3 PAR_IDLE OR CONSTANT **SPOST3**
SPST4 PAR_IDLE OR CONSTANT **SPOST4**
SPST5 PAR_IDLE OR CONSTANT **SPOST5**
SPST6 PAR_IDLE OR CONSTANT **SPOST6**
SPST7 PAR_IDLE OR CONSTANT **SPOST7**
SPST8 PAR_IDLE OR CONSTANT **SPOST8**
SPST9 PAR_IDLE OR CONSTANT **SPOST9**

(TSTATES ARE FINAL STATES FOR CLOCKING TEST DATA)
PAR_IDLE SCK_HI OR SW_HI OR PIX_LO OR LEN_HI OR FEN_HI OR DCK_HI
OR CONSTANT **TSTATE0**

(PLEASE DOWNLOAD KEOCMD FILE)

8.7 KEOcmd7   Advanced Technologies Code

(KEOCMD FORTH CODE FOR KEO CAMERA COMMANDS)

(CONSTANTS FOR COMMAND VECTORS)
92 CONSTANT CMD_STATES
93 CONSTANT CMD_FORMAT
94 CONSTANT CMD_CLEAR
95 CONSTANT CMD_READ
96 CONSTANT CMD_TEST

(FORTH WORDS TO HANDLE STATE DOWNLOADING)

```
: INIT-STATES
   CR
   DSP-WAIT ." DSP READY " CR
   CLR-TXH-TXM
   CMD_STATES CVR!
   SPRE0  DSP16!  SPRE1  DSP16!  SPRE2  DSP16!  SPRE3  DSP16!
   SPRE4  DSP16!  SPRE5  DSP16!  SPRE6  DSP16!  SPRE7  DSP16!
   SPRE8  DSP16!  SPRE9  DSP16!

   SREAD0  DSP16!  SREAD1 DSP16!  SREAD2 DSP16!  SREAD3 DSP16!
   SREAD4 DSP16!  SREAD5 DSP16!  SREAD6 DSP16!  SREAD7 DSP16!
   SREAD8 DSP16!  SREAD9 DSP16!

   SPOST0 DSP16!  SPOST1 DSP16!  SPOST2 DSP16!  SPOST3 DSP16!
   SPOST4 DSP16!  SPOST5 DSP16!  SPOST6 DSP16!  SPOST7 DSP16!
   SPOST8 DSP16!  SPOST9 DSP16!

   PSTATE0 DSP16!  PSTATE1 DSP16!  PSTATE2 DSP16!  PSTATE3 DSP16!
   PSTATE4 DSP16!  PSTATE5 DSP16!  PSTATE6 DSP16!  PSTATE7 DSP16!
   PSTATE8 DSP16!

   TSTATE0 DSP16!
   ." STATES DOWNLOADED" CR ;
```

(FORTH WORDS TO HANDLE FORMAT)
VARIABLE SLENGTH
VARIABLE PLENGTH
VARIABLE SBINNUM
VARIABLE PBINNUM
VARIABLE SPRESCAN
VARIABLE SPOSTSCAN

VARIABLE **PPRESCAN**
VARIABLE **PPOSTSCAN**
VARIABLE **SREADLEN**
VARIABLE **PREADLEN**

.

(INITIALIZE FORMAT PARAMETERS TO DEFAULTS)
DECIMAL
516 SLENGTH !
516 PLENGTH !
1   SBINNUM !
1   PBINNUM !
20 SPRESCAN !
0   SPOSTSCAN !
0   PPRESCAN !
0   PPOSTSCAN !
516 SREADLEN !
516 PREADLEN !

: **INIT-FORMAT**
   DSP-WAIT
   CLR-TXH-TXM
   CMD_FORMAT CVR!
   SLENGTH @ DSP16!
   PLENGTH @ DSP16!
   SBINNUM @ DSP16!
   PBINNUM @ DSP16!
   SPRESCAN @ DSP16!
   SPOSTSCAN @ DSP16!
   PPRESCAN @ DSP16!
   PPOSTSCAN @ DSP16!
   SREADLEN @ DSP16!
   PREADLEN @ DSP16!
   ." FORMAT INITIALIZED " CR ;

(WORDS TO DEAL WITH THE COUNTER, USEFUL FOR TIMING THINGS)

(TWAIT WAITS A GIVEN NUMBER OF COUNTER TICKS.)
(EXPECTS THAT NUMBER ON THE STACK)

CODE **TWAIT**
POPD, TCNT ADDD, BEGIN, TCNT CPD, .CS. UNTIL., NEXT JMP, END-CODE

DECIMAL

```
: MWAIT   ( W - ) ( WAIT W MILLISECONDS )
     0 DO 1870 TWAIT LOOP ;

: DWAIT   ( W - ) ( WAIT W DECISECONDS )
     0 DO 100 MWAIT LOOP ;

: SWAIT   ( W - ) ( WAIT W SECONDS )
     0 DO 1000 MWAIT LOOP ;



( WORDS TO DEAL WITH KEO'S CUSTOM I/O INTERFACE BOARD )
HEX

( RAM LOCATION TO STORE LAST BYTE SENT TO IO CONTROL REGISTER )
VARIABLE   RAMCTL

B400 CONSTANT IOCTL    ( I/O CONTROL REGISTER LOCATION )

: CTL!   ( B - ) ( WRITES A BYTE TO THE IO CONTROL REGISTER )
     DUP RAMCTL C! IOCTL C! ;

0 CTL!    ( INIT IO TO ZERO , IS THIS OK ? )


( WRITING TO BITS 0 1 AND 2 SET THE FILTER WHEEL POSTION )
: FW_SET              ( DESIRED-POSTION -- ERROR FLAG, 0 = OK, OTHER = ERROR )
     07 AND                  ( MASK OFF OTHER BITS                )
     RAMCTL C@ OR            ( MASK IN OTHER BITS WE ALREADY SET  )
     DUP CTL!                ( STORE IT                           )
     1 SWAIT                 ( WAIT FOR IT TO SWITCH              )
     RAMCTL C@ = NOT ;       ( CHECK IT                           )


HEX

: LATCH!  D800 C! ;

: OPEN    FF LATCH! ;
: CLOSE   00 LATCH! ;



( WORDS TO SET THE INTENSIFIER GAIN )
HEX
```

```
: INT-GAIN-SET    ( DESIRED-GAIN -- ERROR FLAG, 0 = OK, OTHER = ERROR )
     18 AND
     RAMCTL C@ OR
     DUP CTL!
     1 SWAIT
     IOCTL C@ = NOT ;


10 CONSTANT LOGAIN
00 CONSTANT MIDGAIN
08 CONSTANT HIGAIN


: LO-GAIN
     LOGAIN PORTA C! ;

: MID-GAIN
     MIDGAIN PORTA C! ;

: HI-GAIN
     HIGAIN PORTA C! ;


: TEST-EN  ( ENABLE TEST DATA THROUGH INTERFACE BOARD )
     PACTL C@ T-7 PACTL C! PORTA C@ T-7 PORTA C! ;

: CAM-EN   ( ENABLE CAMERA DATA THROUGH INTERFACE BOARD )
     PACTL C@ T-7 PACTL C! PORTA C@ F-7 PORTA C! ;


: SHADE    ( GENERATE A TEST PATTERN )
     TEST-EN  DSP-WAIT CMD_TEST CVR! DSP-WAIT ;

: READ     ( READ OUT THE CCD )
     CAM-EN   DSP-WAIT CMD_READ CVR! DSP-WAIT ;

: CLEAR    ( CLEAR THE CCD ONCE )
     DSP-WAIT CMD_CLEAR CVR! DSP-WAIT ;

: BIAS     ( READ OUT A BIAS FRAME )
     CLOSE  CLEAR CLEAR CLEAR READ ;


( WORDS TO PERFORM DARK FRAME ACQUISITION )
```

233

```
: MDARK   ( W - )   ( W MILLISECOND DARK )
    CLOSE CLEAR MWAIT READ ;

: DDARK   ( W - )   ( W DECISECOND DARK )
    CLOSE CLEAR DWAIT READ ;

: SDARK   ( W - )   ( W SECOND DARK )
    CLOSE CLEAR SWAIT READ ;


( WORDS TO PERFORM EXPOSURE FRAME ACQUISITION )

: MOBS   ( W - )   ( W MILLISECOND OBSERVATION )
    CLOSE CLEAR OPEN MWAIT CLOSE 1 DWAIT READ ;

: DOBS   ( W - )   ( W DECISECOND OBSERVATION )
    CLOSE CLEAR OPEN DWAIT CLOSE 1 DWAIT READ ;

: SOBS   ( W - )   ( W SECOND OBSERVATION )
    CLOSE CLEAR OPEN SWAIT CLOSE 1 DWAIT READ ;


( WORDS THAT ARE USEFUL FOR TESTING HARDWARE )
: SHADES   BEGIN SHADE ?TERMINAL UNTIL ;
: READS    BEGIN READ  ?TERMINAL UNTIL ;
: CLEARS   BEGIN CLEAR ?TERMINAL UNTIL ;

. INIT   DSP-BOOT SR DSP-DUMP INIT-STATES INIT-FORMAT MID-GAIN ;

( STORE THE AUTOSTART SEQUENCE AND THE CFA OF THE AUTOSTART WORD IN
EXTERNAL EPROM )

EPROM CONSTANT ASTART

: ASTART!
    EEUNPROT                    ( UNPROTECT THE EEPROM               )
    A44A DUP ASTART EE-!        ( STORE THE AUTOSTART PATTERN        )
    [ ' RESTORE ] LITERAL CFA   ( GET THE CFA OF OUR AUTOSTART WORD  )
    DICT-START -                ( DICT OFFSET OF AUTOSTART WORD      )
    EEDICT-START +              ( LOCATION OF AUTOSTART WORD IN EEPROM )
    DUP ASTART 2+ EE-!          ( STORE THAT AFTER THE AUTOSTART PATTERN )
    EEPROT                      ( PROTECT THE EEPROM                 )
    CR ." AUTOSTART SEQUENCE STORED " CR ;


( END OF KEO CAMERA CODE )
( Please LOAD the KEO Consultants Code: MIP*.FOR now... )
```

234

8.8 FORTH Directory Contents   C:\MIP\FORTH

8.8.1 Advanced Technologies FORTH Code

The following are the FORTH files provided by Advanced Technologies for their controller board. These files are listed above in Sections 8.2 through 8.7.

keoadc7	1901	9/30/91	1:55:40pm
keoadc8	1901	4/13/92	2:29:48pm
keoasm7	5916	9/30/91	10:52:58am
keoasm8	5902	8/24/92	11:06:12am
keocmd7	6047	8/24/92	12:23:02pm
keodsp7	7210	9/30/91	10:54:04am
keorom7	6626	9/30/91	10:54:38am
keotim7	6954	9/30/91	10:55:22am

8.8.2 Advanced Technologies DSP Code

The following are the DSP56001 files provided by Advanced Technologies. These files are all the files associated with the development of the downloaded DSP code and are provided for reference only. The DSP Assembler and Simulator programs are MacIntosh based. All development for the DSP code was done on Macs.

intequ.asm	1112	9/30/91	10:56:54am
ioequ.asm	8978	9/30/91	10:57:16am
keo7.asm	16106	9/30/91	10:58:04am
keo8.asm	16391	10/2/91	4:11:18pm
keo7.dsp	2999	9/30/91	10:56:30am
keo8.dsp	3164	10/2/91	4:12:28pm
keo7.lod	3046	9/30/91	10:58:46am
keo8.lod	3206	10/2/91	4:11:46pm
keo7.lst	32540	5/22/91	9:51:50am
keo8.lst	34831	10/2/91	4:11:46pm

INTEQU.ASM:	DSP Interrupt Equates	(provided by Motorola)
IOEQU.ASM:	DSP I/O Equates	(provided by Motorola)
KEO*.ASM	DSP Assembler code	(developed by Adv. Tech)
KEO*.LOD	DSP Binary code -- outputted from DSP Assembler	
KEO*.LST	DSP code Listing -- outputted from DSP Assembler	
KEO*.DSP	DSP Binary code -- reformatted to be downloaded to controller	
	/* This is the file that is used with the DNLD command */	

8.8.3 KEO Consultants FORTH Code

The following list the main controller FORTH code written by KEO Consultants for use with the MIP and HAARP imagers. Versions are chronological and all versions are provided for reference. The latest version number should always be the version downloaded on the the controller's EEPROM. The code listing for MIP10.FOR is provided in Section 8.1

☐ mip01.for	3306	5/22/91	6:04:20pm
☐ mip02.for	6471	5/28/91	5:19:22pm
☐ mip03.for	8810	7/1/91	12:48:42pm
☐ mip04.for	10842	10/13/91	12:42:20pm
☐ mip05.for	11781	12/20/91	5:14:20pm
☐ mip06.for	12089	7/1/92	11:46:54am
☐ mip07.for	12424	9/15/92	12:23:08pm
☐ mip08.for	14687	10/13/92	10:30:04am
☐ mip09.for	15573	10/19/92	3:26:28pm
☐ mip10.for	16054	7/23/93	11:47:54am

8.8.4 KEO Consultants FORTH Utilities

The following utilities were written by KEO Consultants to assist in the development and testing of the camera electronics and control system:

        TESTFW.FOR
        TESTSH.FOR
        TESTADC4.FOR

These utilities can be downloaded to the controller by sending them as a text file using *Microcsoft*'s TERMINAL program provided with Windows 3.1. When opening TERMINAL, open the HAARP.TRM configuration file. To get to the FORTH environment (terminating the MIP command LOOP), type a '0<cr>'. The FORTH identifier should appear. Hit <cr> a few times to confirm communication. The FORTH system should return an 'OK' for each <cr> entered.

TESTFW.FOR: Contains utilities to test the performance of the filterwheel. There are quite a few dictionary words defined in this file but most are words that are used in the two main functions:

        <ddd> TEST_FILPOS   <ddd> = repetition time in deciseconds on stack

        <ppp> TIME_FW   <ppp> = target position (1-5) on stack

TEST_FILPOS sets up a loop to automatically test all filter move posibilities. Each move returns a status line that gives the MOVE type, the TARGet position, the number of

20msec intervals it took to make the move or the ERRor return code, the PRESent position, and the cycle number. This program is terminated by hitting any key. It will take up to 5 moves to terminate, however. Upon termination, TEST_FILPOS will report all the error tallies for this session. An example of running this program would be:

```
50 TEST_FILPOS
MV 1+ TARG: 3 #: 28 PRES: 3 NUM: 1
MV 2- TARG: 1 #: 37 PRES: 1 NUM: 2
MV 1- TARG: 5 #: 21 PRES: 5 NUM: 3
MV 2+ TARG: 2 #: 36 PRES: 2 NUM: 4
MV 1- TARG: 1 #: 26 PRES: 1 NUM: 5
MV 1+ TARG: 2 TO TE PRES: 4 NUM: 6
MV 2- TARG: 2 TO TE PRES: 4 NUM: 7
MV 1- TARG: 3 #: 25 PRES: 3 NUM: 8
MV 2+ TARG: 5 #: 38 PRES: 5 NUM: 9
MV 1- TARG: 4 #: 27 PRES: 4 NUM: 10

NUMBER OF MOVES: 10
NUMBER OF TIMEOUTS: 2
NUMBER OF POS. ERRS: 0
NUMBER OF TARGET ERRORS. 2
OK
```

The above example ran the TEST_FILPOS utilities using a 5 second cycle time, and 10 moves were executed in which there were two TO (timeout) and two TE (target) errors. These were forced by setting the filterwheel control panel to manual and forcing a move to filter #4. The program waited 1 second for the filter to move from #1 to #2 (during which it moved .o #4), and reported a time out error, and also a target error, because the Present filter #4 is not equal to the Target filter #2. A Position error occurs when the position read from the camera controller is not a legal position. Normally, during execution of TEST_FILPOS, there should be no errors of any kind.

TIME_FW lets the user execute a single move and check the number of loop cycles it took to reach the target filter position. The RET: code is given to check the execution of the filter move. In the example below, we can see that we moved from some unknown position to filter #4, from #4 to #2, from #2 to #3, and from #3 to #5.

```
4 TIME_FW INDEX: 27  RET: 4 OK
2 TIME_FW INDEX: 35  RET: 2 OK
3 TIME_FW INDEX: 26  RET: 3 OK
5 TIME_FW INDEX: 36  RET: 5 OK
```

The time it took for a move is INDEX x .02 seconds. Thus:

Mv: -2   .70 seconds
Mv: +1   .52 seconds
Mv: +2   .72 seconds

The file TESTSH.FOR has utilities to test the operation of the camera's shutters. In the file are the following utilities:

<decisecond> TEST_SH	Opens and closes each shutter sequentially
<decisecond> TEST_BT	Opens and closes both shutters simultaneously
<decisecond> TS1	Opens and closes Shutter1 (front)
<decisecond> TS2	Opens and closes Shutter2 (back)
<decisecond> LGSH	Opens and closes shutters sequentially and logs results

All of these utilities are called with the cycle time in deciseconds on the stack. The first two utilities test both shutters terminating if an error occurs. Testing the shutters both simultaneously and sequentially test the robustness of the shutter power-supply and the shutters' reliability under both conditions. TS1 and TS2 are used to test an individual shutter for possible problems and terminate if an error occurs. LGSH is used to test both shutters for long-term reliability, reporting the results of a long test upon termination of the test (by hitting any key). A few examples follow:

```
30 TEST_BT
ERROR OPENING SH2:
PROGRAM TERMINATED AFTER 3 ITERATIONS OK

30 TS1
ERROR OPENING SH1:
PROGRAM TERMINATED AFTER 5 ITERATIONS OK

30 LGSH
ITER 1 ERR1: 0 ERR2: 0
ITER 2 ERR1: 0 ERR2: 0
ITER 3 ERR1: 0 ERR2: 0
ITER 4 ERR1: 0 ERR2: 0
ITER 5 ERR1: 0 ERR2: 0
ITER 6 ERR1: 0 ERR2: 0
ITER 7 ERR1: 0 ERR2: 1
ITER 8 ERR1: 0 ERR2: 1
ITER 9 ERR1: 1 ERR2: 1
ITER 10 ERR1: 2 ERR2: 1
ITER 11 ERR1: 2 ERR2: 1
ITER 12 ERR1: 2 ERR2: 1
ITER 13 ERR1: 2 ERR2: 1
ITER 14 ERR1: 2 ERR2: 1

PROGRAM TERMINATED AFTER 14 ITERATIONS
SH1 ERRORS: 2   SH2 ERRORS: 1 OK
```

In the above examples, errors were forced by setting the shutters OPEN or CLOSED manually via the camera's control panel.

The file TESTADC4.FOR contains utilities for testing the camera's Analog to Digital Converter interface. The 68HC11 has an 8 channel multiplexed ADC port that is used by

238

the camera to read 4 temperatures and a photodiode output. The following utilities facilitate testing and calibrating these circuits:

<ADC Channel> RD_CH   Reads a channel four times and takes an average

```
1 RD_CH
1: 192 2: 192 3: 192 4: 192 TOTAL: 768 AVERAGE 192
```

<ADC Channel> TST_RD   Executes RD_CH once a second

```
1 TST_RD
1: 191 2: 191 3: 192 4: 191 TOTAL: 765 AVERAGE 191
1: 191 2: 191 3: 192 4: 191 TOTAL: 765 AVERAGE 191
1: 192 2: 192 3: 191 4: 192 TOTAL: 767 AVERAGE 191
CH:1 OK
```

<10 µsec interval> RD-AV   Reads Photodiode channel and gives an average. The average defined here is to read the channel four times, take an average, wait <int>x10µsec, take another four readings, average, and do this four times.

```
100 RD-AV
AVERAGE: 25
```

<10 µsec interval> TST-AV   Reads Photodiode every second giving the average

```
100 TST-AV
AVERAGE: 26
AVERAGE: 26
CH: 100 OK
```

<10 µsec interval> AUTO-AV Reads Photodiode every second using the µsec interval to read it four times. The average is computed. If a key is hit, the µsec interval is incremented and the averaging continues. To terminate the program, a key must be hit twice

```
25 AUTO-AV
INTERVAL: 25
AVERAGE: 25
AVERAGE: 25
AVERAGE: 25

INTERVAL: 26
AVERAGE: 26
AVERAGE: 25
AVERAGE: 25

INTERVAL: 27
AVERAGE: 26
AVERAGE: 25
```

```
AVERAGE:  24

INTERVAL:  28
AVERAGE:  24
AVERAGE:  26
AVERAGE:  25
AVERAGE:  26
OK
```

The AUTO-AV utility is used to check for and optimize the averaging period for reading the photodiode voltage. A reminder for the ADC channels:

CH0:	CCD Temperature
CH1:	TEC Temperature
CH2:	Intensifier Temperature
CH3:	Filter Wheel Temperature
CH4:	Photodiode Temperature

8.8.5 Compress Utility and LINKLST.FOR

COMPRESS.EXE is a DOS application that can be run on any FORTH text file. This code was developed by Advanced Technologies and strips the text file of any unnecessary characters thereby speeding up the downloading time. A 'compressed' version of the file is created:

MIP10.FOR ----> MIP10.CMP

Downloading MIP10.CMP will create exactly the same code as MIP10.FOR.

LINKLST.FOR was a FORTH development idea to create separate links in the FORTH dictionary that made it impossible for the user to use any FORTH words, except those defined after the Link was broken. The link could later be restored. This utility was developed with the thought of having two different 'environments' available to the user depending on their FORTH skill level. It was later decided not to use this approach, but the code has been left on the HAARP machine for the interested *FORTHIE*!

240

Chapter 9   KEO Hardware Schematics

MIP BLOCK DIAGRAM

KEO CONSULTANTS   CAL

SEPT. 1991

242

KEO Consultants -- Interface Board Layout

Project: MIP Interface Card
Date: 6/12/91
Title: Intensifier Cntrl
Page: 3 of 7
Engineer: C.Lance

SCALE: 72%

Project: MIP Interface Card
Title: FilterWheel Cntrl
Date: 6/12/91
Page: 4 of 7
Engineer: C.Lance
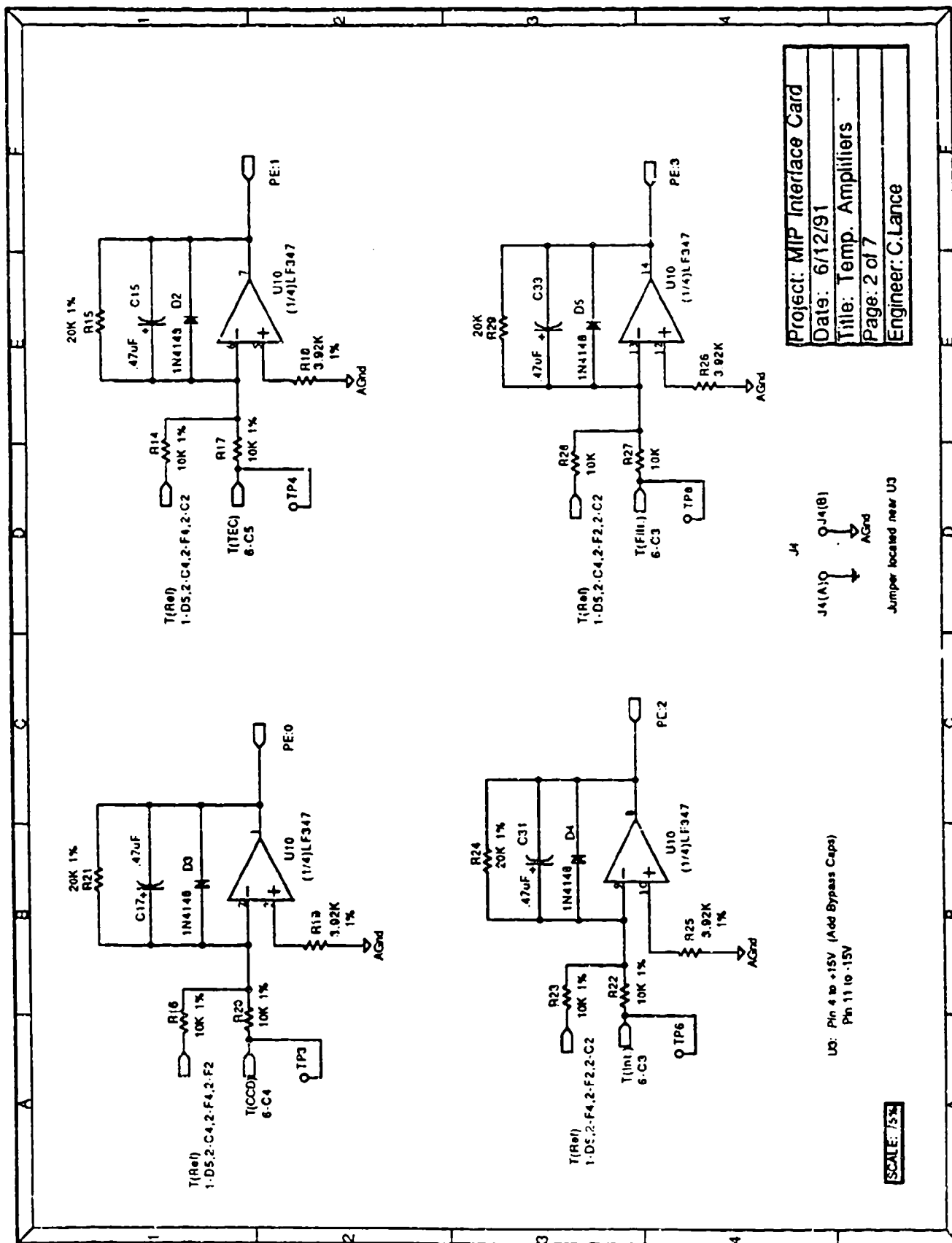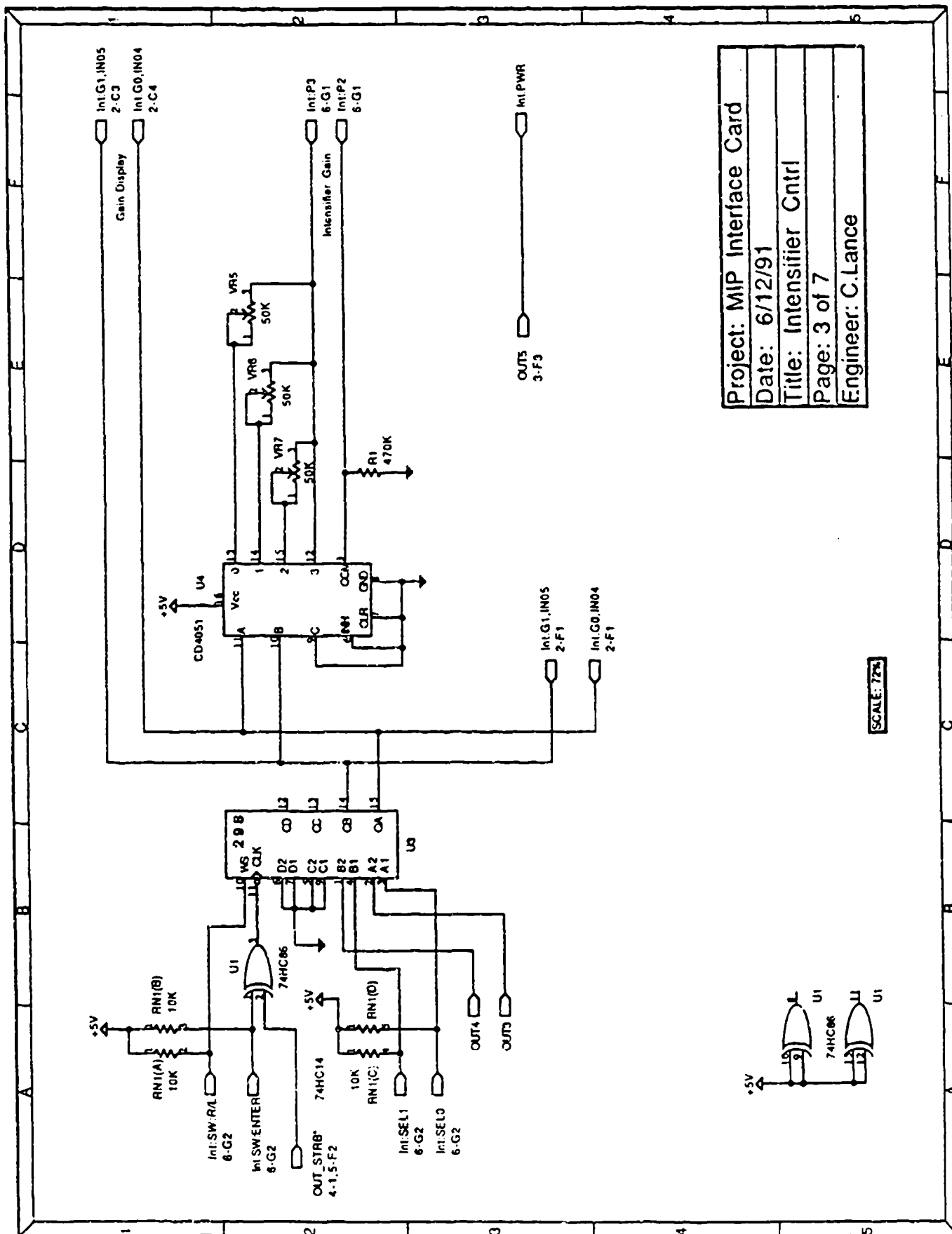
Project: MIP Interface Card
Title: GAL I/O Interf.
Date: 6/12/91
Page: 5 of 7
Engineer: C.Lance
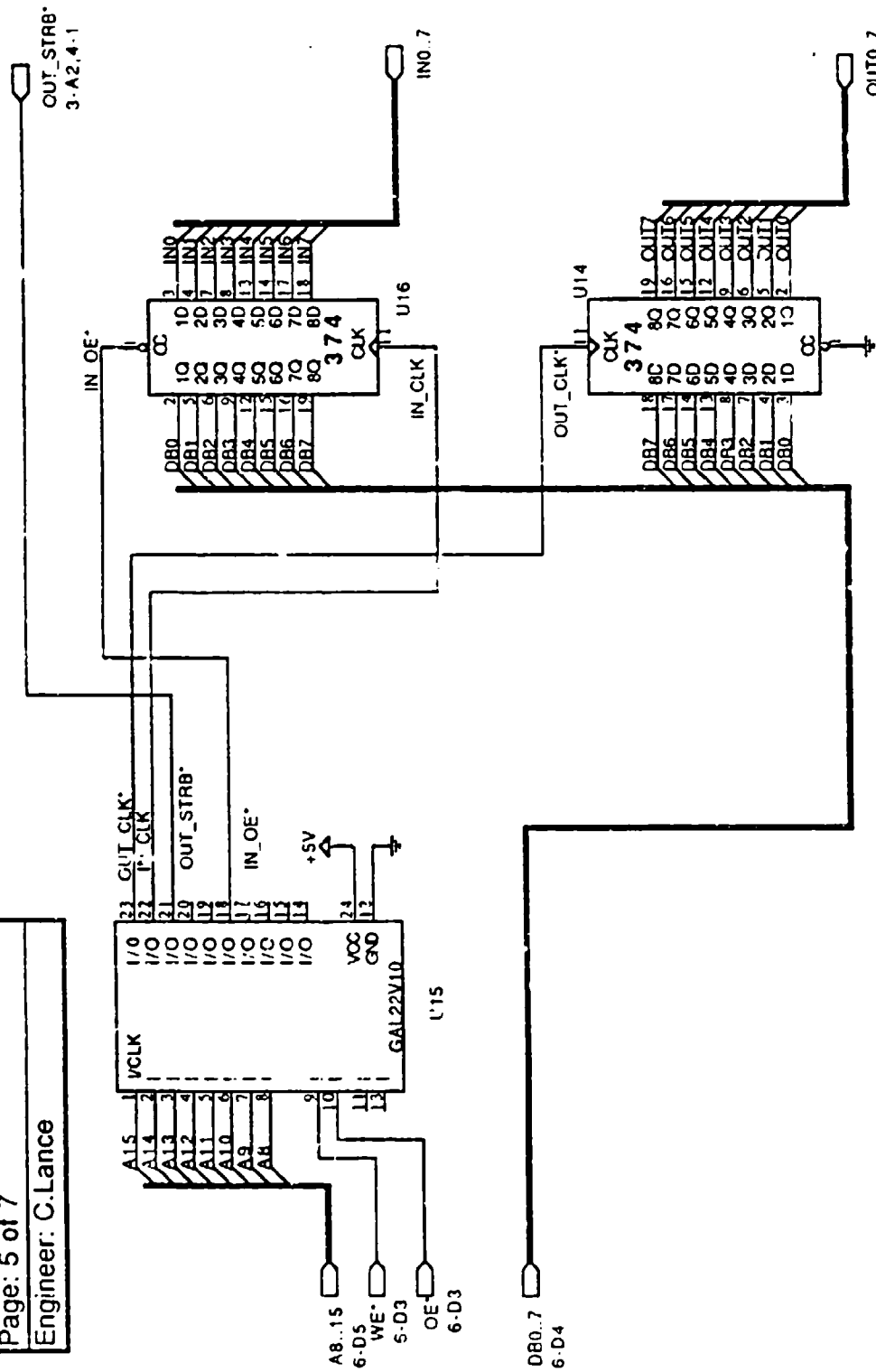
SCALE: 90%

248
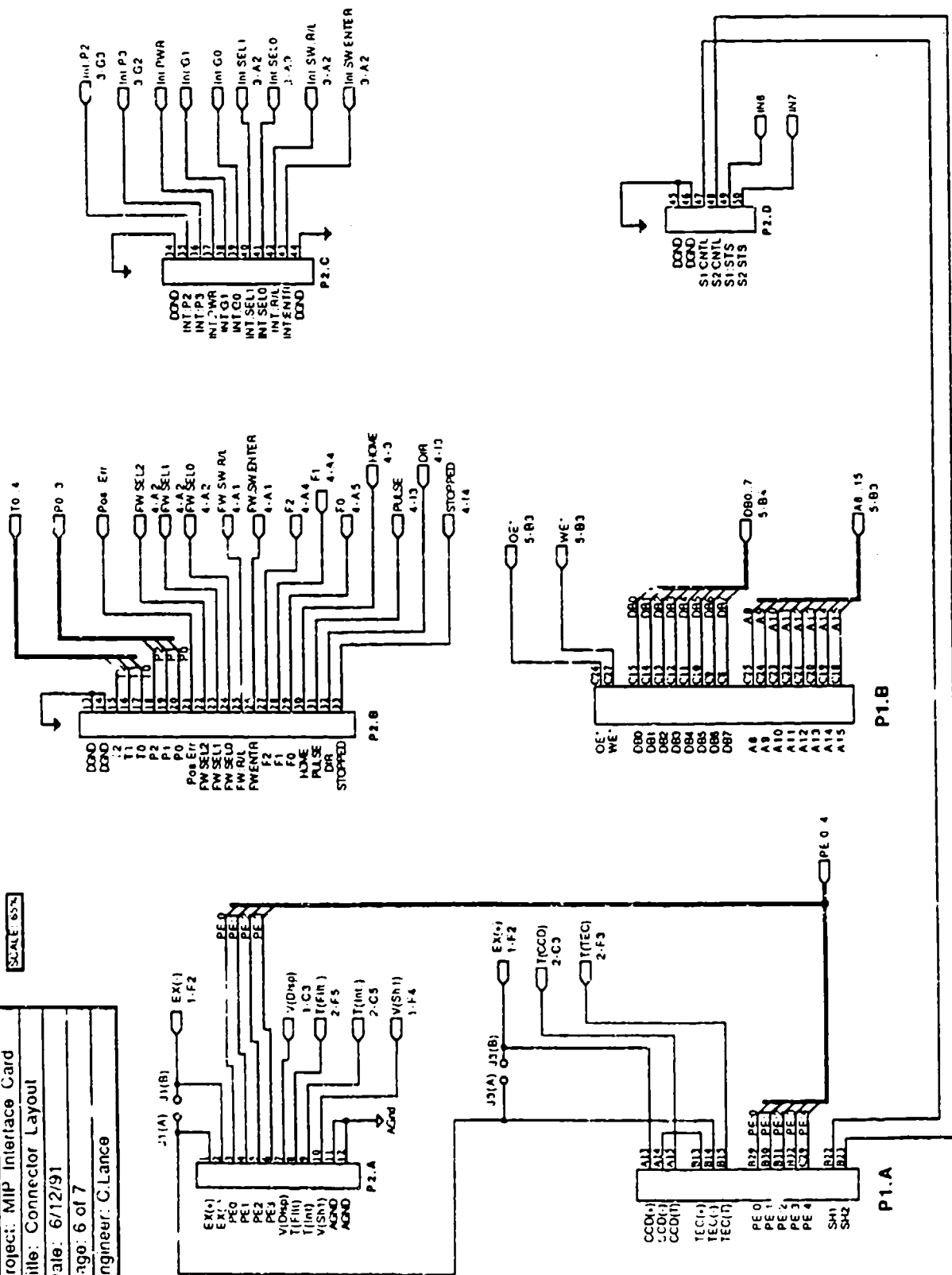
Project: MIP Interface Card
Title: Power
Date: 6/12/91
Page: 7 of 7
Engineer: C.Lance

Tantalum Caps

Monolithic Ceramic Caps

Scale: 85%

P1.C

250

251

```
DEVICE     KEO_IO (pal22v10)

PIN
           a15       =       1 (CLK_INPUT combinatorial)
           a14       =       2 (INPUT combinatorial)
           a13       =       3 (INPUT combinatorial)
           a12       =       4 (INPUT combinatorial)
           a11       =       5 (INPUT combinatorial)
           a10       =       6 (INPUT combinatorial)
           a09       =       7 (INPUT combinatorial)
           a08       =       8 (INPUT combinatorial)
           /we       =       9 (INPUT combinatorial )
           /oe       =      10 (INPUT combinatorial )
           /temp_oe  =          17 (IO combinatorial active_low)
           /in_oe =        18 (IO combinatorial active_low)
           /tmp1_strobe  =      19 (IO combinatorial active_low)
           /tmp2_strobe  =      20 (IO combinatorial active_low)
           /out_strobe   =      21 (IO combinatorial active_low)
           in_clk =        22 (IO combinatorial active_high)
           /out_clk  =          23 (IO combinatorial active_low)

;


BEGIN

"Enable all final outputs"
ENABLE (in_oe);
ENABLE (in_clk);
ENABLE (out_clk);
ENABLE (out_strobe);


" This PAL is used to generate clock signals for a pair of "
" I/O latches on the KEO interface card and is designed as "
" per KEO's spec. "
" The address of the latches is $B4xx "


" in_clk is a rising edge that latches data into the input latch "
in_clk = a15 * /a14 * a13 * a12 * /a11 * a10 * /a09 * /a08 * oe ;

" temp_oe  is the an internal output enable signal for the input latch "
temp_oe = in_clk ;

" in_oe  is a delayed version of temp_oe just to be safe "
in_oe = temp_oe ;

" out_clk is a rising edge that latches data into the output latch "
out_clk = a15 * /a14 * a13 * a12 * /a11 * a10 * /a09 * /a08 * we ;

" KEO also needs a strobe occurring after the output latch has "
" been written to and the data has stabilized. To get this, without "
" using a digital delay line, we will rely on the propagation delay "
" through the PAL 3 times, which should be about 15 ns each time. "
" This will give a delay of approx. 45 ns "

" tmp1_strobe is a delayed version of out_clk "
tmp1_strobe = out_clk ;

" tmp2_strobe is a delayed version of tmp1_strobe "
tmp2_strobe = tmp1_strobe ;

" out_strobe is a delayed version of tmp2_strobe "
"           It will occurr approx 45 ns after out_clk '
"           This is the signal to use as output strobe "
out_strobe = tmp2_strobe ;
```

```
Device Map from File [keo_io] for PLD [pal22v10]*
F0*
L0044 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L0088 0111 1011 0111 0111 1011 0111 1011 1011 1011 1111 1111 *
L0440 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L0484 0111 1011 0111 0111 1011 0111 1011 1011 1111 1011 1111 *
L0924 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L0968 1111 1111 1111 1110 1111 1111 1111 1111 1111 1111 1111 *
L1540 1111 1111 1111 1111 1110 1111 1111 1111 1111 1111 1111 *
L2200 1110 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L2904 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L2948 1111 1111 1111 1111 1111 1111 1110 1111 1111 1111 1111 *
L3696 1111 1101 1111 1111 1111 1111 1111 1111 1111 1111 1111 *
L5808 0111 0101 0101 01 *
C3A13*
9308
```

CY545 EEPROM Program   Rev 3:2
(Half-Stepping Mode)

Address		Command	Comment
	0	^RV	AutoStart Code
	3	O 42H<cr>	Operating Mode (1200 Baud)
	9	"<cr><lf>KEO PROGRAM 08-19-93<cr><lf>	
	34	REV 3:2 C LANCE<cr><lf>"	
	52	/B 4<cr>	Turn USRB4 Off (Enable Motor)
	57	F 360<cr>	First=360 (fast rate for Home)
	63	R 230<cr>	Rate=230 (Peak Velocity)
	69	S 245<cr>	Slope=248 (Acceleration)
1$:	75	H 0<cr>	Home: Look at USRB0
	79	F 19<cr>	First=22 (Slow rate for moves)
	84	B 4<cr>	Turn USRB4 On (Disable Motor)
	88	"W<cr><lf>"	Wait Identifier
2$:	93	T 01H,093<cr>	Loop and test for USRB1 (Move)
	103	T 2,170<cr>	Test USRB2: If set, jmp to 4$: '+'
	111	"-<cr><lf>"	'-' Identifier
	116	-<cr>	Set to CCW direction
	118	T 3,149<cr>	Test USRB3: If set, jmp to 3$: 1 Filter
	126	N 1600<cr><lf>"	'1600 Steps' Identifier
	136	N 1610<cr>	NumSteps = 1610: Move past Home
	143	J 223<cr>	Jmp to 6$: GO command
3$:	149	"N 800<cr><lf>"	'800 Steps' Identifier
	158	N 810<cr>	NumSteps = 810: Move past Home
	164	J 223<cr>	Jmp to 6$: GO command
4$:	170	'+<cr><lf>"	'+' Identifier
	175	+<cr>	Set to CW direction
	177	T 3,208<cr>	Test USRB3: If set, jmp to 5$: 1 Filter
	185	"N 1600<cr><lf>"	'1600 Steps' Identifier
	195	N 1590<cr>	NumSteps = 1590: Move almost to Home
	202	J 223<cr>	Jmp to 6$: GO command
5$:	208	"N 800<cr><lf>"	'800 Steps' Identifier
	217	N 790<cr>	NumSteps = 790: Move almost to Home
6$:	223	/B 4<cr>	Turn USRB4 Off (Enable Motor)
	228	G<cr>	GO command: Make move!
	230	F 360<cr>	First = 360: Fast Rate for Home
	236	J 75<cr>	Jump to 1$: Home command

EPROM Program   27C16:

Location:	Output:
11H	0H
12H	3H
13H	7H
14H	5H
15H	1H
21H	1H
22H	0H
23H	3H
24H	7H
25H	5H
31H	5H
32H	1H
33H	0H
34H	3H
35H	7H
41H	7H
42H	5H
43H	1H
44H	0H
45H	3H
51H	3H
52H	7H
53H	5H
54H	1H
55H	0H

Bit:	0	1
B0:	No Move	Move
B1:	+	-
B2:	200	400

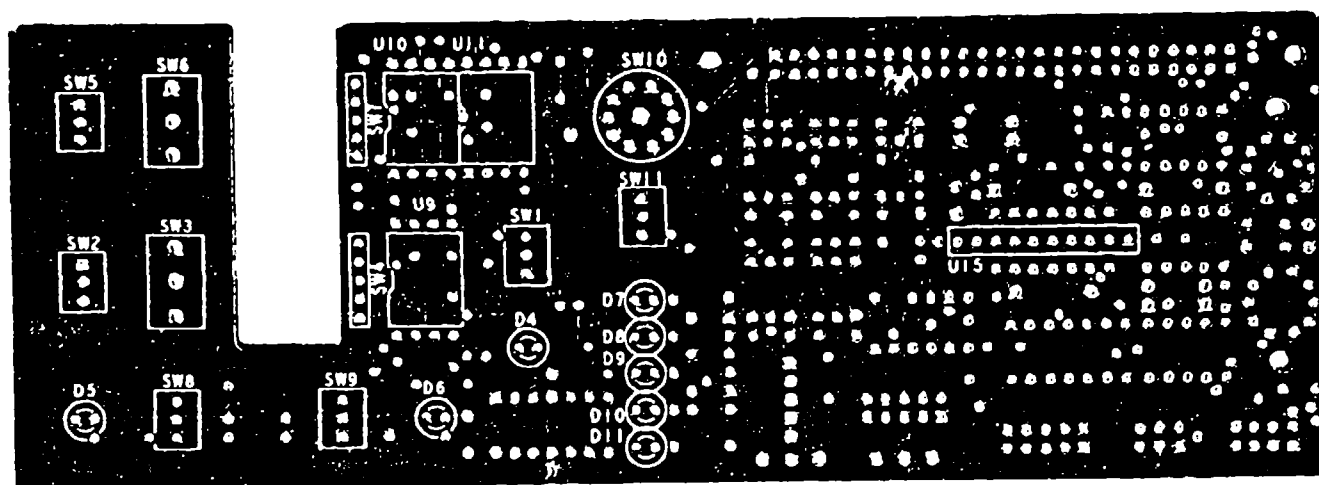| Output:	Commands:
0H	No Move
1H	Move 200 steps +
3H	Move 200 steps -
5H	Move 400 steps +
7H	Move 400 steps -

All other locations in 27C16 set to 8H
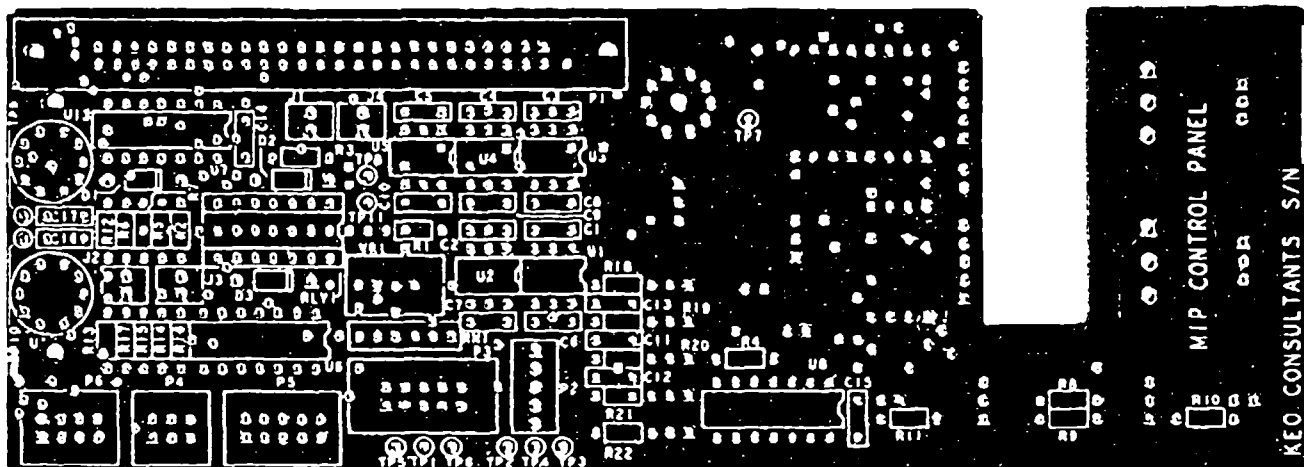
Connector Side   Front
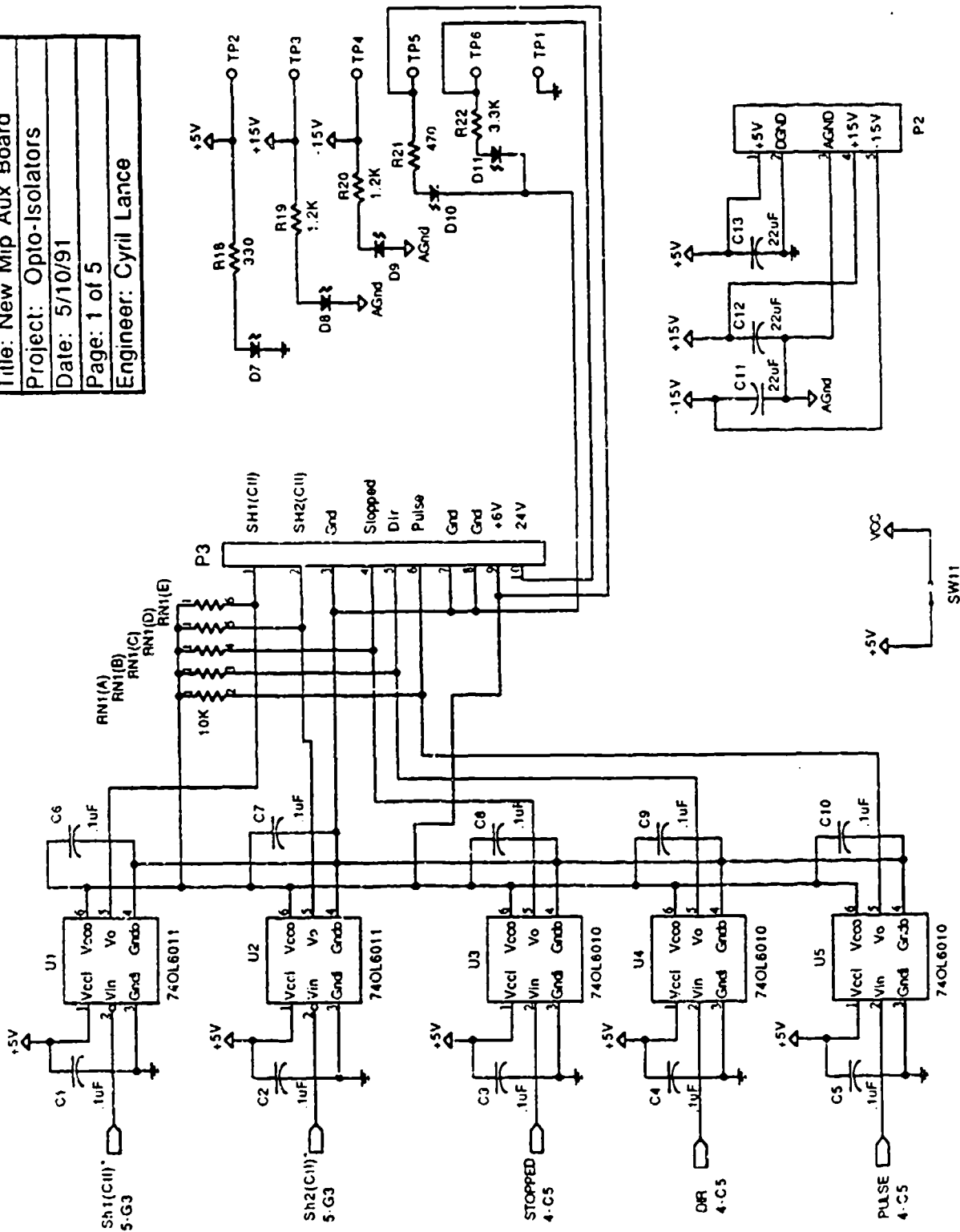
Control Panel Side   Back

MIP Control Panel Layout

tes: U13 and U14 are mounted on the back of the board due to layout error.

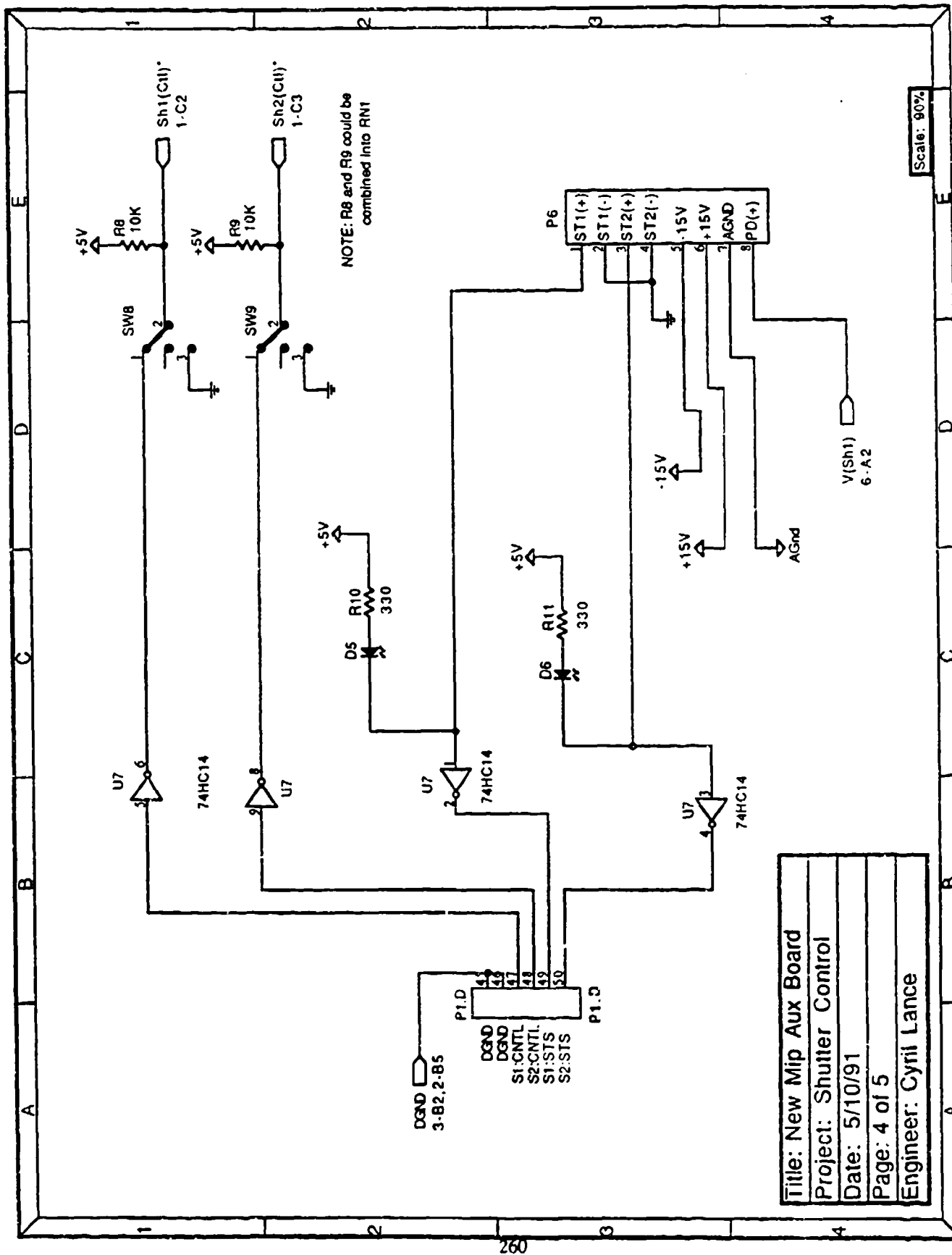Pins 5 and 6 of U6 need to be reversed due to schematic error.

| Title: New Mip Aux Board |
| Project: Opto-Isolators |
| Date: 5/10/91 |
| Page: 1 of 5 |
| Engineer: Cyril Lance |

SCALE 60%

SCALE: 70%

P4

Int:P1
Int:P2
Int:P3
Int:P4
5:RTD+
6:RTD-

Int:RTD(-)

Int:RTD(+)
8-C2

R5
5
AGnd

R8
100
AGnd

U8
(1/4)LF347

R7
10K

AGnd

D4

NOTE: On circuit board
Pins 5 & 6 of U8 were
layed out incorrectly,
so look out for modification
on printed circuit board

SW4

Gnd

RLY1

PRMA100G7

D3
1N4001

AGnd

+5V

R3
36

D2
1N746
3.3V

AGnd

VCC

VCC

LE

Gnd

U9

CP

5082-7300

74HC08

U8

74HC14
U7

D1
4.7V
AGnd

R4
10K

U6
(1/4)LF347

R2
10K

VR1
1M

R1
2K

AGnd

SW1

+5V

SW2

SW3

74HC08

U8

U9

TP1

PC(-)
4-F5

P1.C

DGND
INT:P2
INT:P3
INT:PWR
INT:G1
INT:G0
INT:SEL1
INT:SEL0
INT:R/L
INTENTR
DGND

P1.C

DGND
3-B2,4-B3

Title: New Mip Aux Board
Project: Filter Wheel Control
Date: 5/10/91
Page: 3 of 5
Engineer: Cyril Lance

Scale: 82%

Sh1(Ctl)*
1-C2

Sh2(Ctl)*
1-C3

+5V R8 10K

+5V R9 10K

NOTE: R8 and R9 could be
combined into RN1

SW8 2
3

SW9 2
3

P6
1 ST1(+)
2 ST1(-)
3 ST2(+)
4 ST2(-)
5 -15V
6 +15V
7 AGND
8 PD(+)

-15V

+15V

AGnd

V(Sh1)
6-A2

+5V
R10 330
D5

+5V
R11 330
D6

U7 6 74HC14

U7 8 U7

U7 2 74HC14

U7 3 74HC14

P1.D
16
13
17
18
19
20
P1.D

DGND
3-B2,2-B5

DGND
DGND
S1:CNTL
S2:CNTL
S1:STS
S2:STS

Title: New Mip Aux Board
Project: Shutter Control
Date: 5/10/91
Page: 4 of 5
Engineer: Cyril Lance

Scale: 90%

Project: New Mip Aux Board
Title: Temperature Control
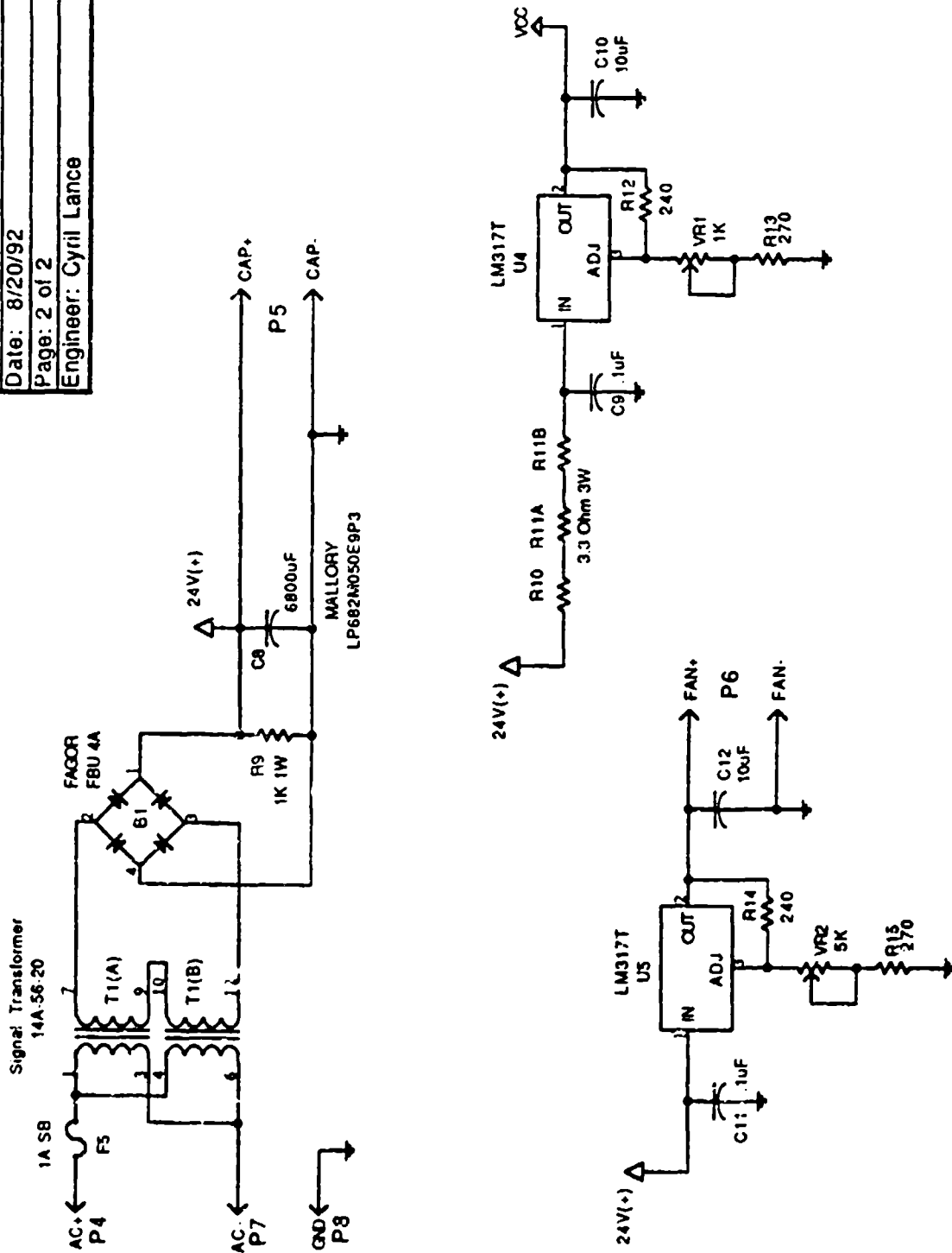Date: 5/10/91
Page: 5 of 5
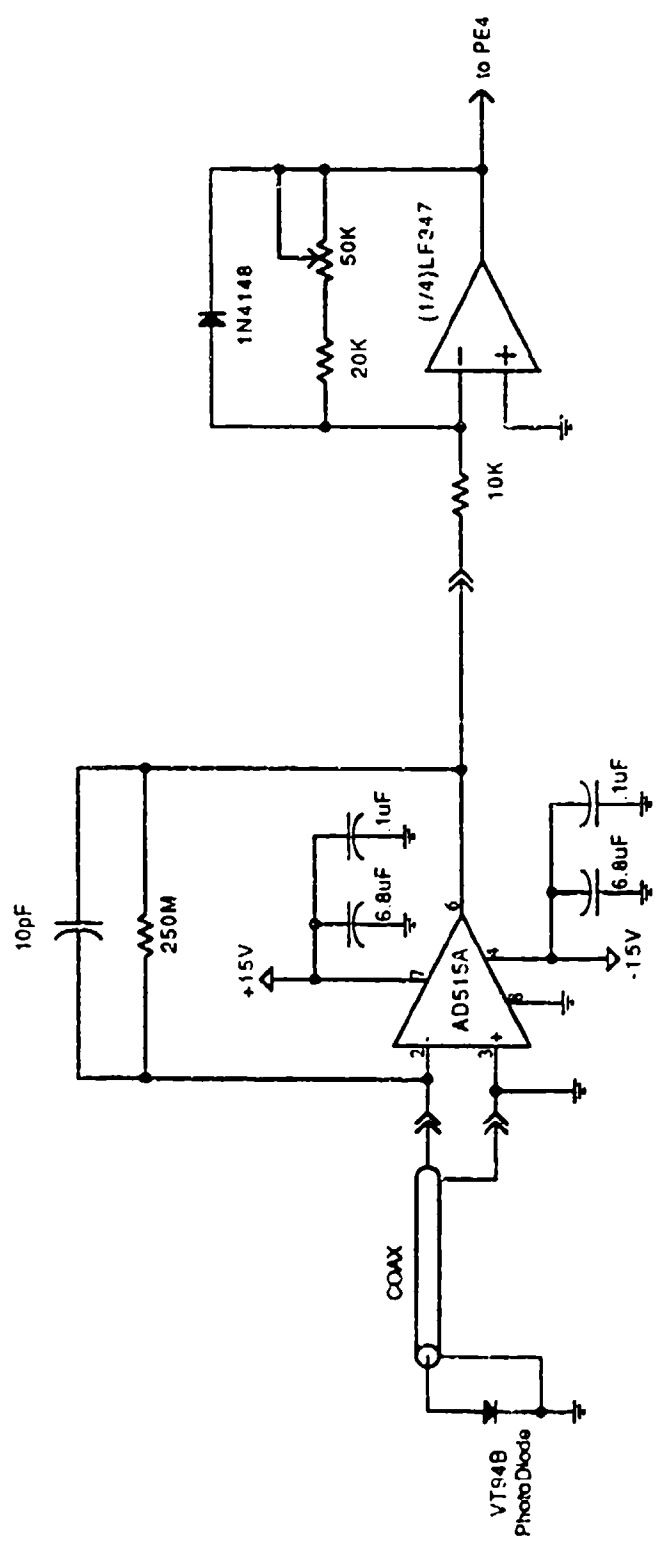Engineer: Cyril Lance
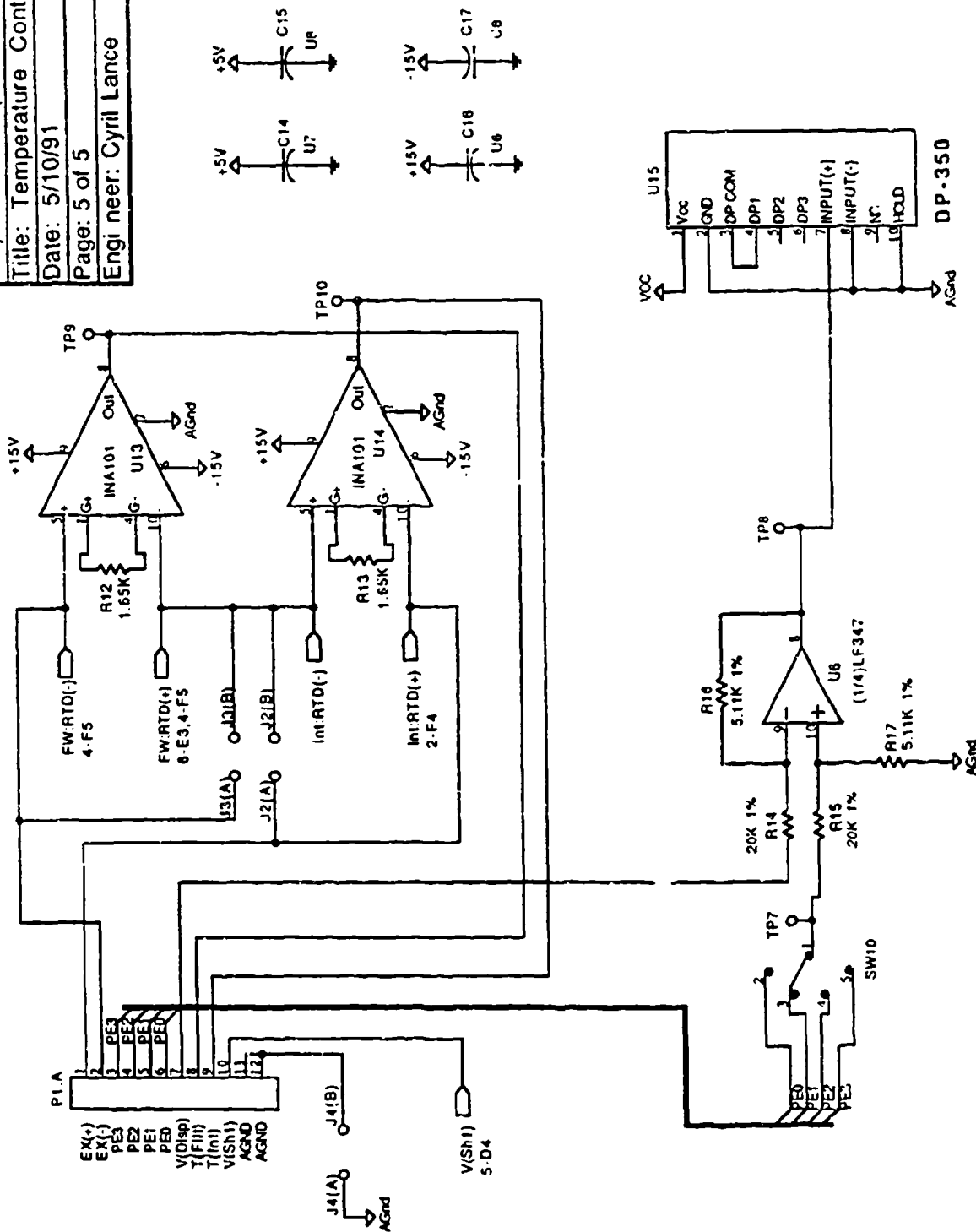
Scale: 80%
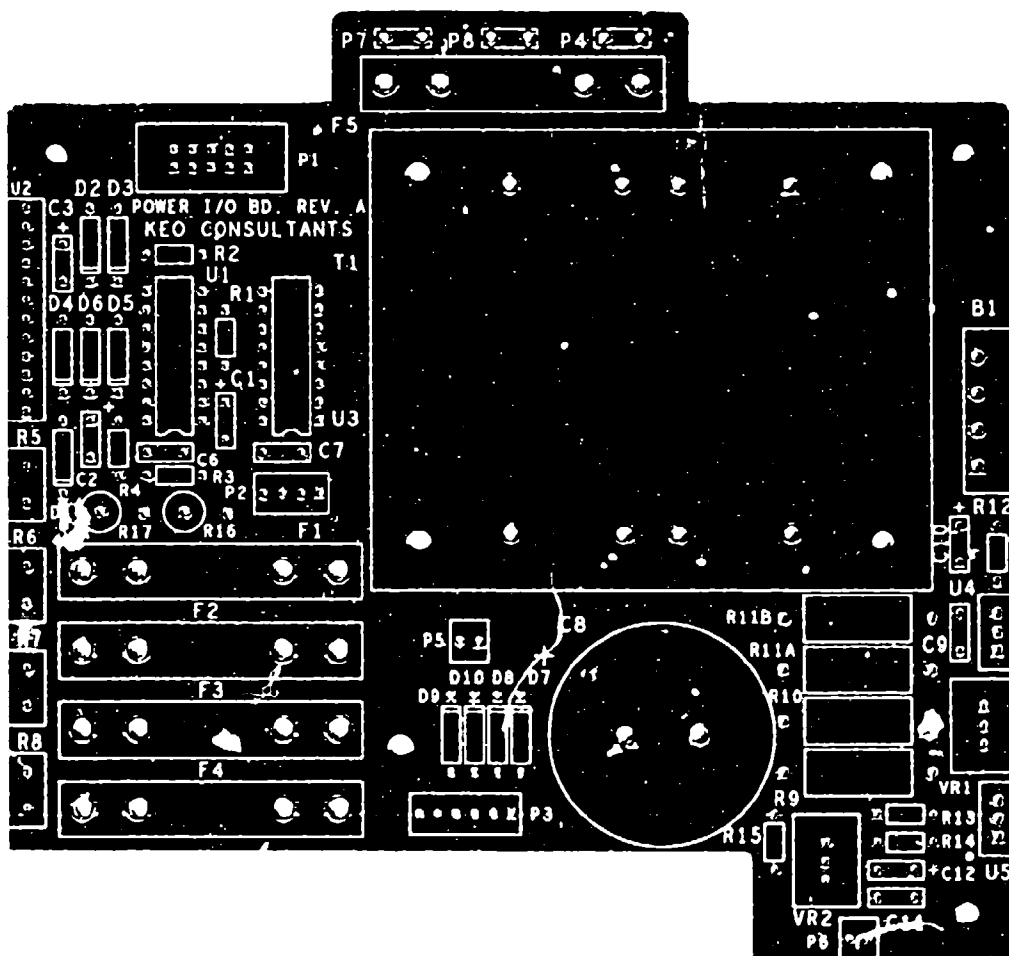
POWER I/O Rev.A Board Layout

263

Project: PowerI/O board- REV.A
Title: 24V Power Supply
Date: 8/20/92
Page: 2 of 2
Engineer: Cyril Lance

Scale: 85%

Project: MIP Camera
PhotoDiode Amplifiers
KEO Consultants
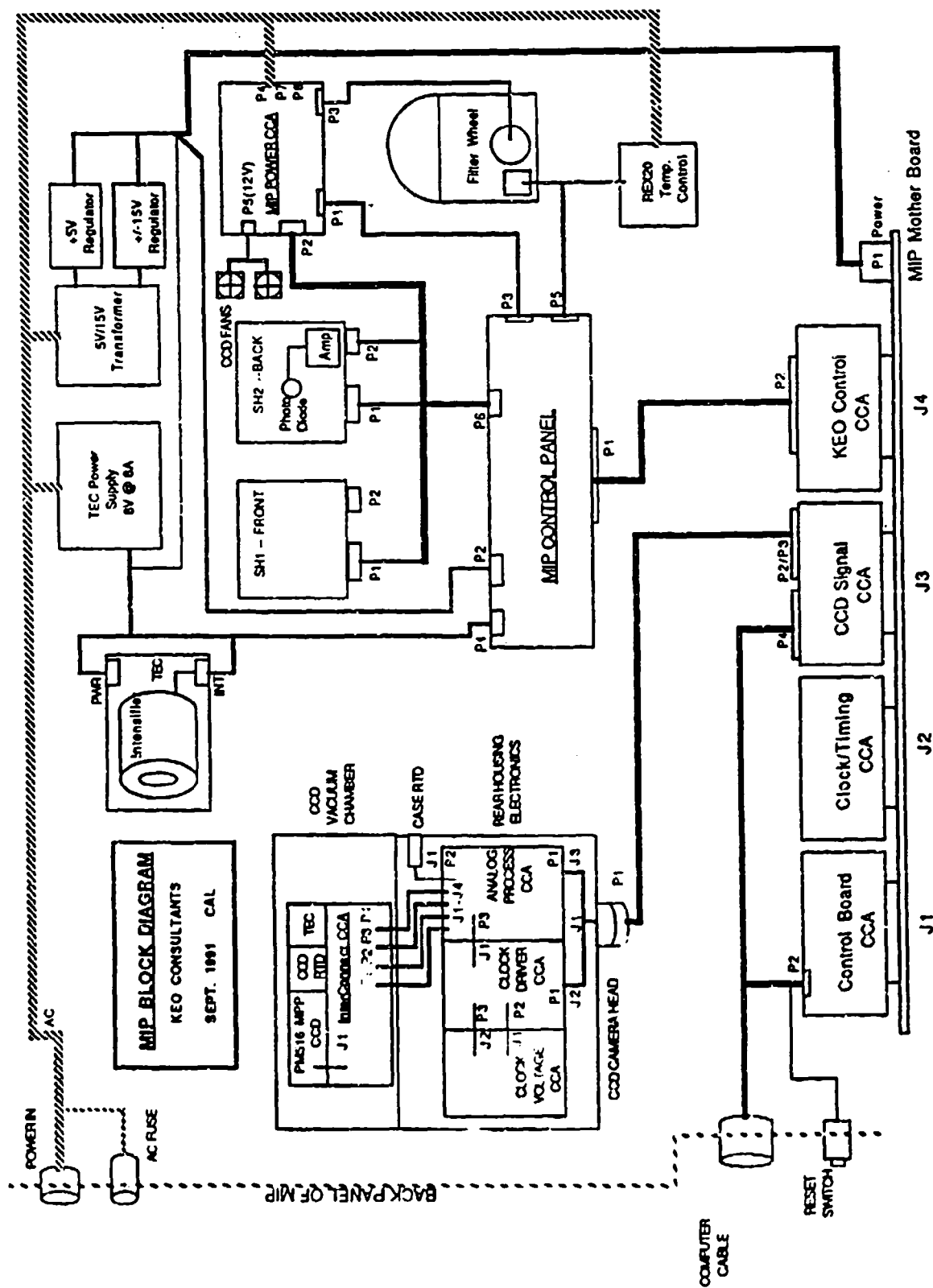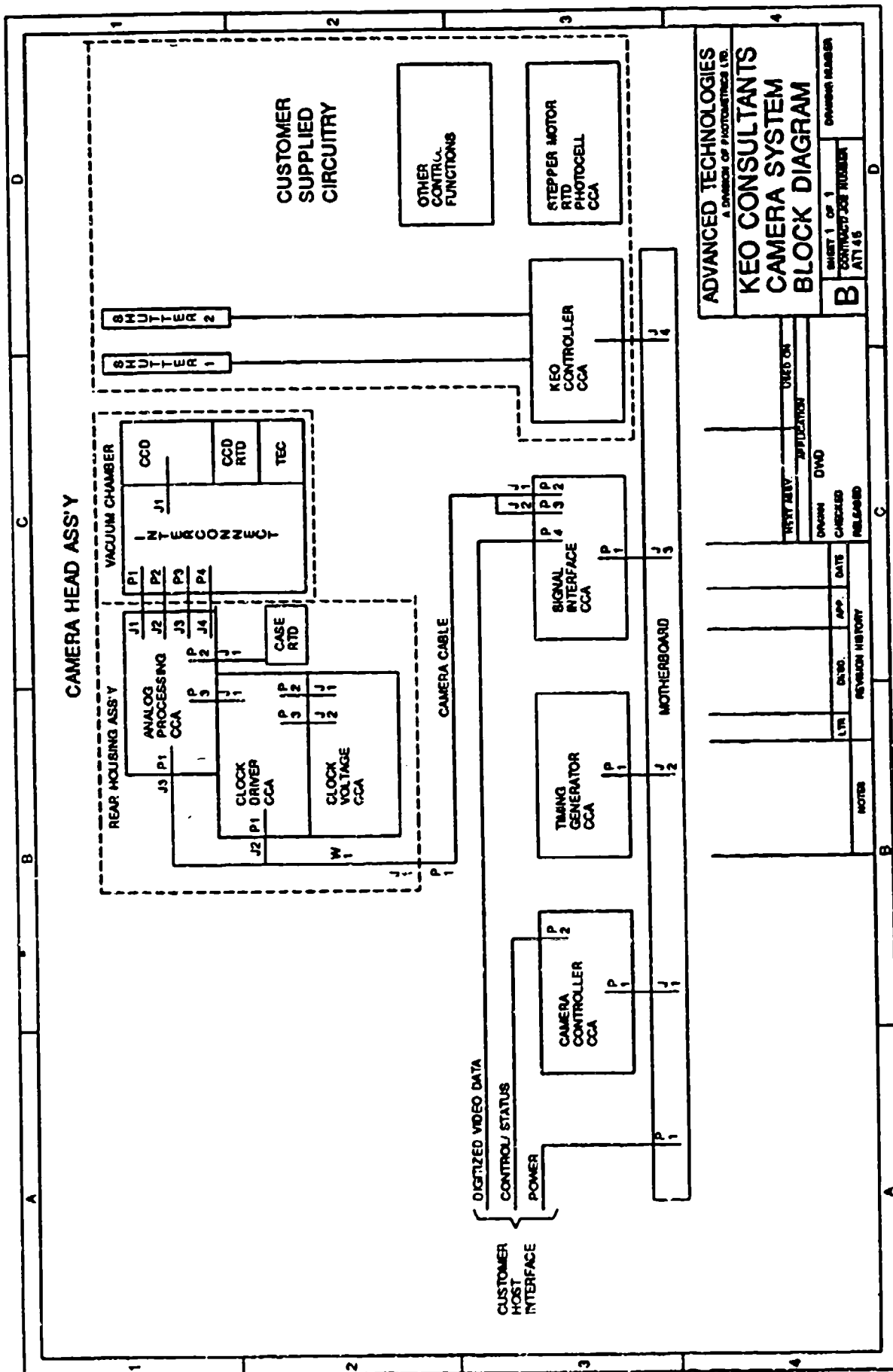Date: 11/90
Eng: Cyril Lance

to PE4

1N4148

50K

(1/4)LF347

20K

10K

Variable Gain Amp on
KEO Interface CCA

10pF

250M

.1uF

6.8uF

.1uF

6.8uF

+15V

AD515A

-15V

PhotoDiode Amplifier Board
located in Shutter 2

COAX

VT94B
PhotoDiode

265

TRANSMISSION SWITCHES HOAR061-351

Project: HAARP
Title: HAARP Filter Wheel
Page: 1 of 1
Date: 8/20/93
Engineer: Lyin Lance

SCALE 65%

Chapter 10   Advanced Technologies Schematics

MIP BLOCK DIAGRAM

KEO CONSULTANTS

SEPT. 1991 CAL

KEO CONSULTANTS
CAMERA SYSTEM
BLOCK DIAGRAM

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

KEO CAMERA MOTHERBOARD

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

SHEET 1 of 3
CONTRACT/JOB NUMBER
AT145

J3 SIGNAL INTERFACE CCA

J1 KEO CNTL INTERFACE CCA

TO SHEET 3

TO SHEET 1

J3 connector (A pins):
A1 TDAT00, A2 TDAT01, A3 TDAT02, A4 TDAT03, A5 TDAT04, A6 TDAT05, A7 TDAT06, A8 TDAT07, A9 TDAT08, A10 TDAT09, A11 TDAT10, A12 TDAT11, A13 SCLK1+, A14 SCLK2+, A15 SCLK3+, A16 SCLK3+, A17 SCMCLK, A18 PCLK1+, A19 PCLK2+, A20 GATECLK+, A21 RS, A22 DCH, A23 RESET, A24 DGND, A25 DGND, A26 VCC, A27 VCC, A28 VCC, A29 DGND, A30 VCC, A31 DGND, A32 VCC

J3 connector (B pins):
B1, B2, B3, B4 PIXELCLK, B5 EOLCLK, B6 EOFCLK, B7, B8, B9, B10 DATACLK, B11, B12, B13 SCLK1-, B14 SCLK2-, B15 SCLK3-, B16 SCMCLK-, B17 PCLK1-, B18 PCLK2-, B19 PCLK3-, B20 GATECLK-, B21 TDSEL, B22, B23, B24, B25 GAINHIGH, B26 GAINLOW/SEL, B27, B28, B29, B30, B31, B32

J3 connector (C pins):
C1 TEC+, C2 TEC+, C3 TEC+, C4 TEC-, C5 TEC-, C6 TEC-, C7, C8, C9, C10 RTDTEC+, C11 RTDCCD+, C12 RTDTEC-, C13 RTDCCD-, C14 TECT, C15 CCDT, C16, C17, C18, C19, C20, C21 AGND, C22 AGND, C23 AGND, C24 AGND, C25 AGND, C26, C27 +15V, C28 +15V, C29 +15V, C30 -15V, C31 -15V, C32 -15V

J1 connector (B pins):
B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11, B12 RTDTEC+, B13 RTDTEC-, B14 RTDTEC-, B15 TECT, B16, B17, B18, B19, B20, B21, B22, B23 SHUTTER2, B24 SHUTTER1, B25, B26, B27, B28, B29 CCDTEMP, B30 TECTEMP, B31 INTENSTEMP, B32 WHEELTEMP

J1 connector (C pins):
C1, C2, C3, C4, C5, C6, C7 AD07, C8 AD06, C9 AD05, C10 AD04, C11 AD03, C12 AD02, C13 AD01, C14 AD00, C15 E, C16 AS, C17 AS, C18 AT1, C19 AT0, C20 AT3, C21 AT2, C22 AT1, C23 AT0, C24 RD, C25 OE, C26 WE, C27, C28, C29 DIODEVOLTS, C30, C31, C32

J1 connector (A pins):
A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12 RTDCCD+, A13 RTDCCD-, A14 CCDT, A15, A16, A17, A18, A19, A20, A21 AGND, A22 +15V, A23 AGND, A24 +15V, A25 DGND, A26 -15V, A27 DGND, A28 -15V, A29 DGND, A30 VCC, A31 DGND, A32 VCC

NEXT ASSY.   USED ON
APPLICATION

DRAWN   DWG
CHECKED
RELEASED
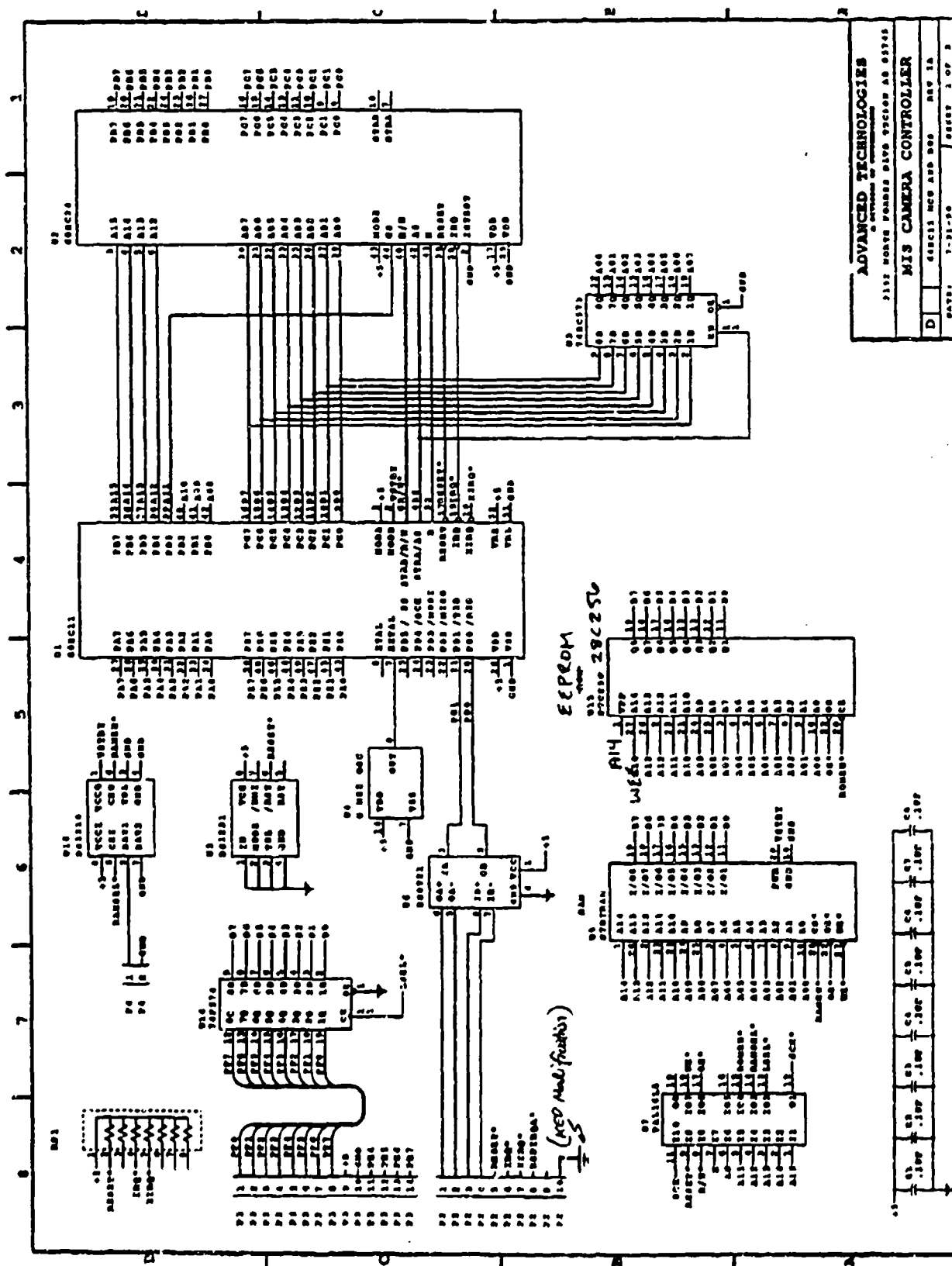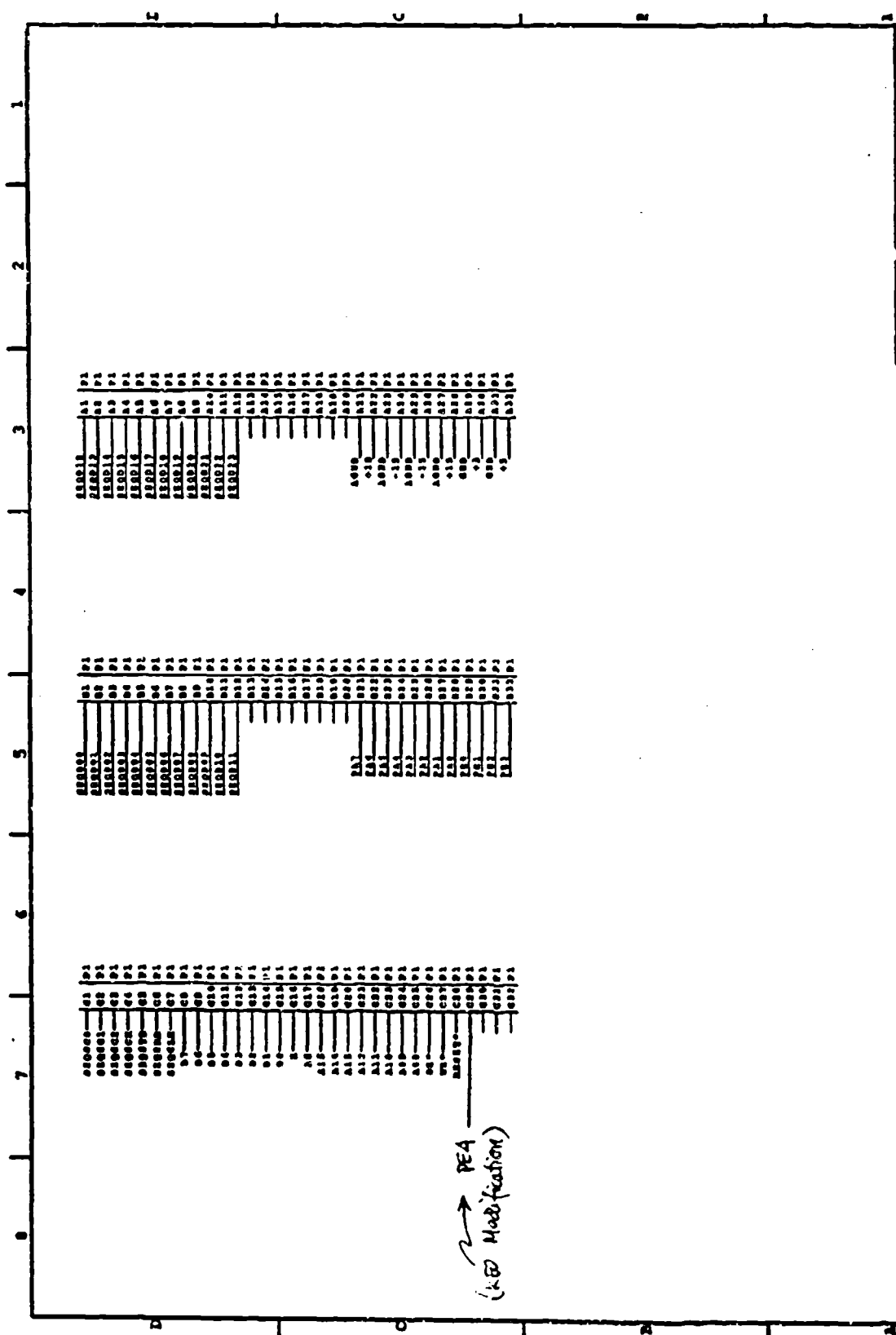
LTR | DESC. | APP. | DATE
REVISION HISTORY

NOTES

TO SHEET 2

P1
1 VCC
2 DGND
3 AGND
4 +15V
5 -15V
6 TEC-
7 TEC-
8 TEC+
9 TEC-

DS1
DS2
DS3

R1 Val
R2 Val
R3 Val

VCC
+15V
-15V
DGND

C10 47uF, 6V
C7 47uF, 6V
C4 47uF, 6V
C1 47uF, 6V
VCC
DGND

C13 22uF, 25V
C12 22uF, 25V
C9 22uF, 25V
C6 22uF, 25V
C3 22uF, 25V
-15V

C11 22uF, 25V
C8 22uF, 25V
C5 22uF, 25V
C2 22uF, 25V
+15V
AGND

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

KEO CAMERA MOTHERBOARD

SHEET 3 OF 3
DRAWING NUMBER
CONTRACT/JOB NUMBER
AT145
A

USED ON
APPLICATION
NEXT ASSY.

DRAWN DWD
CHECKED
RELEASED

LTR DESC. APP. DATE
REVISION HISTORY

NOTES

ADVANCED TECHNOLOGIES
MIS CAMERA CONTROLLER

```
DEVICE mis_addr_select_2 (pal1618)

PIN
    a15 =         1 (INPUT combinatorial)
    a14 =         2 (INPUT combinatorial)
    a13 =         3 (INPUT combinatorial)
    a12 =         4 (INPUT combinatorial)
    a11 =         5 (INPUT combinatorial)
    as  =         6 (INPUT combinatorial)
    e   =         7 (INPUT combinatorial)
    rd  =         8 (INPUT combinatorial)
    /reset =      9 (INPUT combinatorial)
    sckin =      11 (INPUT combinatorial)
    /sckout =    12 (OUTPUT combinatorial active_low)
    /lsel   =    13 (OUTPUT combinatorial active_low)
    /ramsel  =   14 (OUTPUT combinatorial active_low)
    /romsel  =   15 (OUTPUT combinatorial active_low)
    /memdis =    16 (IO combinatorial active_low)
    /oe  =       17 (OUTPUT combinatorial active_low)
    /we  =       18 (OUTPUT combinatorial active_low)

;

BEGIN

"Enable all outputs"
ENABLE (sckout);
ENABLE (lsel);
ENABLE (ramsel);
ENABLE (romsel);
ENABLE (memdis);
ENABLE (oe);
ENABLE (we);

"ramsel =  select lower 32k"
ramsel =   /a15 * /reset * /memdis;

"romsel =  select upper 32K"
romsel = a15 * /reset * /memdis;

"lsel = latch select "
lsel = a15 * a14 * /a13 * a12 * a11 * /rd * e * /reset;

"disable memory at DAC and internal register locations"
memdis = ( a15 * /a14 * a13 * a12 * /a11 ) +    "68HC24 internal registers"
         ( a15 * a14 * /a13 * a12 * a11 ) +     "LATCH (lsel) location"
         (reset);                               "disable on reset"

"output enable"
oe = rd * e * /reset;

"write enable"
we = /rd * e * /reset;

" Just an inverter"
sckout = sckin;

END.
```
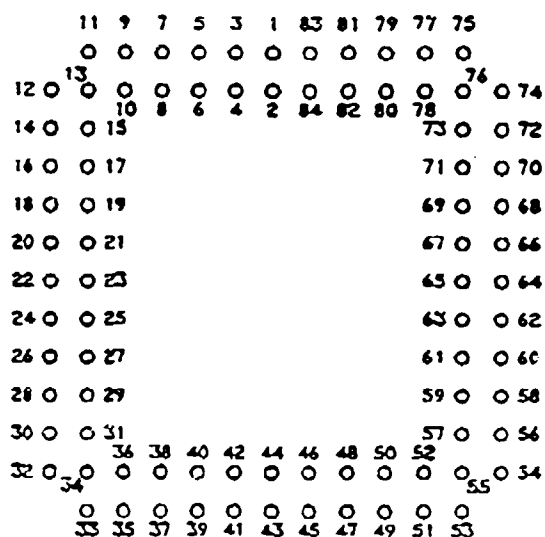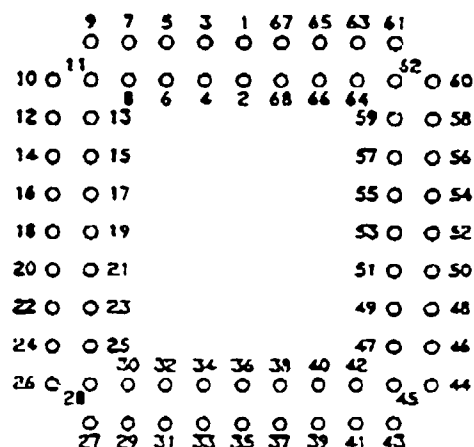
RECOMMENDED P. C. BOARD PATTERNS
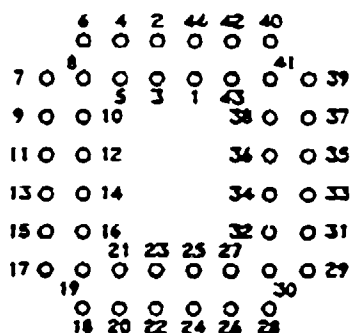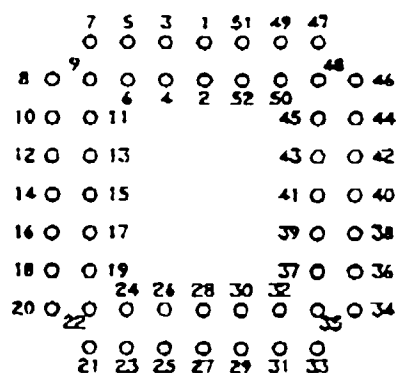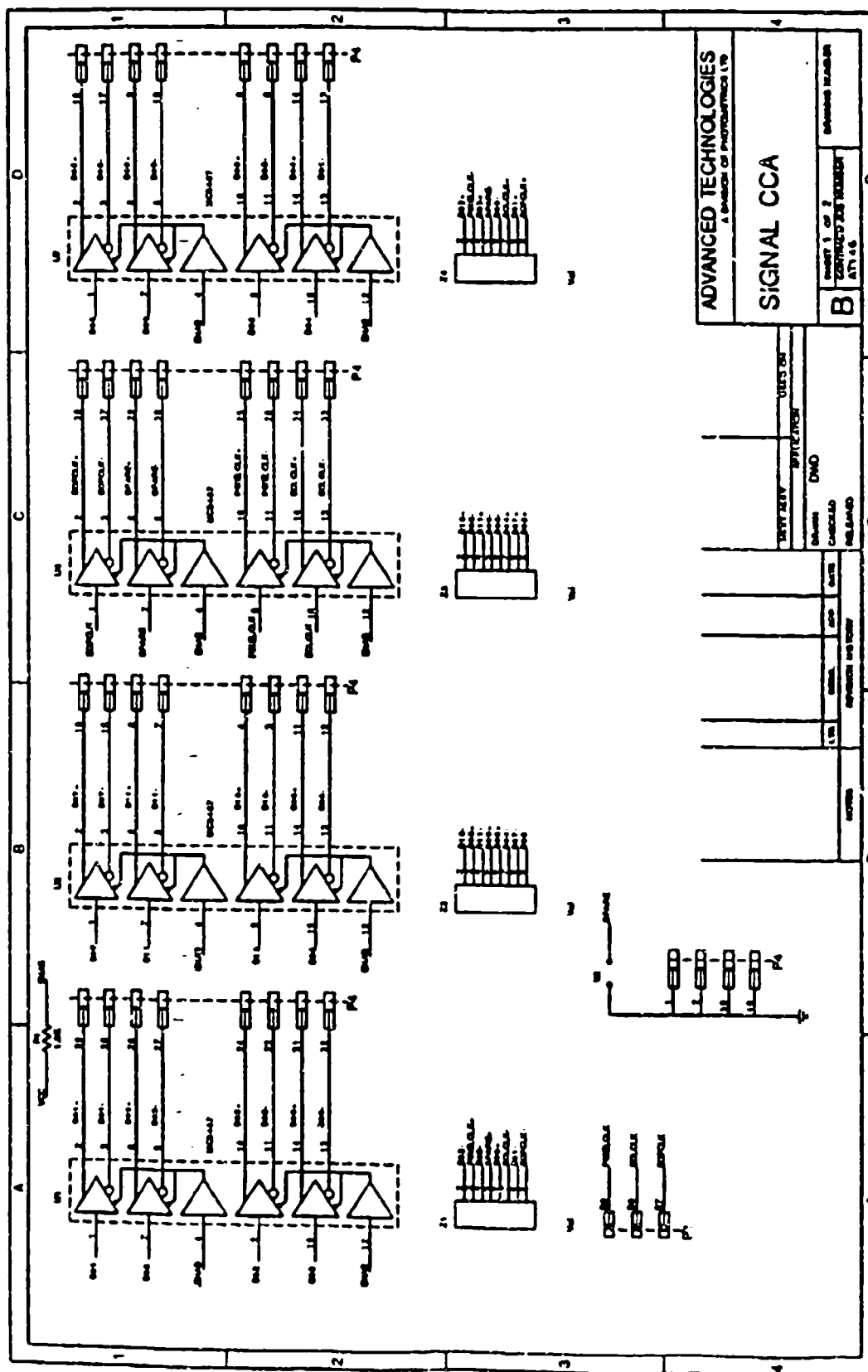# AND NUMBERING SEQUENCE

84 POS.

68 POS.

44 POS.

52 POS.

ADVANCED TECHNOLOGIES
A Division of Photometrics Ltd

SIGNAL CCA

SIGNAL CCA

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

CLOCK TIMING GENERATOR

280

DRIVER CCA

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD

SHEET 2 OF 4

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

DRIVER CCA

SHEET 5 OF 4
CONTRACT 205 NUMBER
ATI 45

ADVANCED TECHNOLOGIES
A division of PHOTOMETRICS LTD.

KEO
ANALOG PROCESSING CCA

CCD DRIVE CONNECTIONS

POWER DISTRIBUTION

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

ANALOG PROCESSING CCA

SHEET 2 OF 2
CONTRACT/JOB NUMBER
AT145

CLOCK POWER CCA

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

SHEET 1 of 2
CONTRACT/JOB NUMBER
AT145
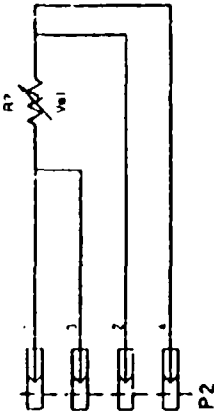
CLOCK POWER CCA

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD

SHEET 2 OF 2
CONTRACT/JOB NUMBER
ATI 45

ADVANCED TECHNOLOGIES
A DIVISION OF PHOTOMETRICS LTD.

CASE TEMPERATURE RTD

DRAWING NUMBER

SHEET OF
CONTRACT/ JOB NUMBER

A

USED ON

APPLICATION

NEXT ASSY

DRAWN
CHECKED
RELEASED

LTR | DESC. | APP. | DATE
REVISION HISTORY

NOTES

R?
Va1

P2

Photometrics Unregulated TEC Supply

291

Chapter 11   Cabling and Internal Wiring

HAARP DC Wiring Diagram

293

HAARP Imager
AC Wiring Diagram

REX C100 Temperature Controller

AC- 7
AC+ 6

Black 20AWG

Transformer

Power-One HBAA-40W-A

1 3 2 4

Wht 20 AWG

Circuit Breakers
Potter-Brumfield
W28-XQ11A-1

FUSE 1A SB
FUSE 1A SB
FUSE 1A SB

Wht 20 AWG

Wht 20 AWG

Photometrics
TEC Power Supply

AC5 AC2
AC3 AC1
GND

Cooling FAN
Pabst Boxer
3108PS-12W-B30

FAN- FAN+

BRN

BLUE

Splice

Aerovox AO6GEN1C13
EMI Filter

BRN
BLUE
GRN/YL

Wht 20 AWG

Black 20AWG

Power I/O Board

AC+ AC-
GND

Back Panel Fuse

FUSE 3A SB

Back Panel On/Off Switch
ALCO SRV-210N

Wht 18AWG

Blk 18AWG

Grn 18AWG

Rear Panel AC Connector

A B C

MIP/HAARP to IBM PC Cable Layout

P1

Male

P1
from MIP/HAARP camera

GND Lug
to Shield

Male

DB44
to AFG Board

Female

DB9
to RS422
port

KEO Consultants

HAARP External Cable Pinout Documentation

Signal Name	From:	Pin#:	To:	Pin#:	Wire pair
DP0+	DB44	13	HAARP.P1	A	green
DP0-	DB44	10	HAARP.P1	B	red
DP1+	DB44	32	HAARP.P1	C	white
DP1-	DB44	31	HAARP.P1	D	red
DP2+	DB44	34	HAARP.P1	E	blue
DP2-	DB44	33	HAARP.P1	F	red
DP3+	DB44	36	HAARP.P1	H	orange
DP3-	DB44	35	HAARP.P1	J	black
DP4+	DB44	38	HAARP.P1	K	brown
DP4-	DB44	37	HAARP.P1	L	black
DP5+	DB44	40	HAARP.P1	M	yellow
DP5-	DB44	39	HAARP.P1	N	red
DP6+	DB44	42	HAARP.P1	P	white
DP6-	DB44	41	HAARP.P1	R	black
DP7+	DB44	7	HAARP.P1	S	red
DP7-	DB44	5	HAARP.P1	T	black
DP8+	DB44	43	HAARP.P1	U	green
DP8-	DB44	9	HAARP.P1	V	black
DP9+	DB44	30	HAARP.P1	W	blue
DP9-	DB44	27	HAARP.P1	X	black
DP10+	DB44	29	HAARP.P1	Y	green
DP10-	DB44	28	HAARP.P1	Z	orange
DP11+	DB44	15	HAARP.P1	a	brown
DP11-	DB44	19	HAARP.P1	b	red
FEN+	DB44	22	HAARP.P1	c	green
FEN-	DB44	12	HAARP.P1	d	blue
LEN+	DB44	23	HAARP.P1	e	green
LEN-	DB44	14	HAARP.P1	f	brown
VSCLK+	DB44	21	HAARP.P1	g	orange
VSCLK-	DB44	11	HAARP.P1	h	red
GND	DB44	1	HAARP.P1	k	white
GND	DB44	44	HAARP.P1	j	blue
RCV+	DB9	9	HAARP.P1	m	green
RCV-	DB9	1	HAARP.P1	n	white
TRN+	DB9	2	HAARP.P1	p	green
TRN-	DB9	3	HAARP.P1	q	yellow
Note: DB9 setup for RS422 boards using					
the B&B Electronics 232CICC pinouts					

MIP/HAARP Cabling Documentation
KEO Consultants

Internal Cable: Rear Panel of Instrument to Interface controller cards

Name	Interface Brd.	Rear Panel conn.	color (&pair)
D0+	31	A	Grn (red)
D0-	32	B	Red (grn)
D1+	36	C	Wht (red)
D1-	35	D	Red (wht)
D2+	24	E	Blue (red)
D2-	23	F	Red (Blue)
D3+	28	H	Org (Blk)
D3-	27	J	Blk (Org)
D4+	14	K	Brn (Blk)
D4-	13	L	Blk (Brn)
D5+	18	M	Yllw (Red)
D5-	17	N	Red (Yllw)
D6+	11	P	Wht (Blk)
D6-	12	R	Blk (Wht)
D7+	16	S	Red (Blk)
D7-	15	T	Blk (Red)
D8+	5	U	Grn (Blk)
D8-	6	V	Blk (Grn)
D9+	9	W	Blue (Blk)
D9-	10	X	Blk (Blue)
D10+	4	Y	Grn (Org)
D10-	3	Z	Org (Grn)
D11+	8	a	Brn (Red)
D11-	7	b	Red (Brn)
PIXEL CLK+	25	g	Org (Red)
PIXEL CLK-	26	h	Red (Org)
EOL CLK+	34	e	Grn (Brn)
EOL CLK-	33	f	Brn (Grn)
EOF CLK+	38	c	Grn (Blue)
EOF CLK-	37	d	Blue (Grn)
GND	1	k	Wht (Blue)
GND	2	j	Blue (Wht)

Name	DSP Cntrl Brd.	Rear Panel conn.	color (&pair)
OA(+)/RTX(+)	1	m	Grn (Wht)
OA(-)/RTX(-)	2	n	Wht (Grn)
IB(+)/TDX(+)	3	p	Grn (Yllw)
IB(-)/TDX(-)	4	·q	Yllw (Grn)
RESET	5	SW(+)	Yllw (Blk)
GND	10	SW(-)	Blk (Yllw)

Connectors:

Rear Panel:

MIP/HAARP Computer Interface Cable: 37 PIN Circular 67 Series Amphenol

Internal 4"x6" boards:

Adv. Tech Signal Board: AMP Mod IV 44 Pin Double Row .1"
Adv. Tech DSP Cntrl Board: AMP Mod IV 10 Pin Double Row .1"

Signal CCA to Camera Head Cable for HAARP

CCD Head	Measured	Signal CCA	Color	Wire Gage
63 PIN Circ.	signal	P2 -24 Pin IDC		
		P3- 40 Pin IDC		
P1-1	AD11	P3-34	GREEN/PURPLE	28 AWG
P1-2	DGND	P2-19	PURPLE/GREEN	28 AWG
P1-3	AD10	P3-20	BROWN/PURPLE	28 AWG
P1-4	GND	P2-19	PURPLE/BROWN	28 AWG
P1-5	AD09	P3-36	BLACK/GREEN	28 AWG
P1-6	DGND	P2-20	GREEN/BLACK	28 AWG
P1-7	AD08	P3-32	WHITE/BROWN	28 AWG
P1-8	DGND	P2-20	BROWN/WHITE	28 AWG
P1-9	AD07	P3-30	YELLOW/GRAY	28 AWG
P1-10	DGND	P2-21	GRAY/YELLOW	28 AWG
P1-11	CCDTEMP	P3-6	PURPLE	24 AWG
P1-12	AD06	P3-18	YELLOW/BROWN	28 AWG
P1-13	DGND	P2-21	BROWN/YELLOW	28 AWG
P1-14	AD05	P3-40	BLUE/WHITE	28 AWG
P1-15	DGND	P2-22	WHITE/BLUE	28 AWG
P1-16	AD04	P3-38	RED/BROWN	28 AWG
P1-17	DGND	P2-22	BROWN/RED	28 AWG
P1-18	AD03	P3-24	YELLOW/ORANGE	28 AWG
P1-19	DGND	P2-23	ORANGE/YELLOW	28 AWG
P1-20	AD02	P3-22	RED/GRAY	28 AWG
P1-21	DGND	P2-23	GRAY/RED	28 AWG
P1-22	AD01	P3-28	GREEN/WHITE	28 AWG
P1-23	DGND	P2-24	WHITE/GREEN	28 AWG
P1-24	AD00	P3-26	WHITE/GRAY	28 AWG
P1-25	DGND	P2-24	GRAY/WHITE	28 AWG
P1-26	DCR	P3-37	GRAY/PURPLE	28 AWG
P1-27	GAINSEL1	P3-7	PURPLE/GRAY	28 AWG
P1-28	ADS	P3-35	ORANGE/PURPLE	28 AWG
P1-29	GAINSEL2	P3-8	PURPLE/ORANGE	28 AWG
P1-30	TEC+	P2-1/P2-2	PURPLE/YELLOW	24 AWG
P1-31	TEC-	P2-3/P2-4	BROWN/BLUE	24 AWG
P1-32	CCDRTD-	P3-4	RED	24 AWG
P1-33	CCDRTD+	P3-2	GREEN	24 AWG
P1-34	TECRTD-	P3-3	BLUE	24 AWG
P1-35	TECRTD+	P3-1	WHITE/BLACK	24 AWG
P1-36	TECTEMP	P3-5	GRAY	24 AWG
P1-37	(+)15V	P2-5	WHITE	24 AWG
P1-38	(-)15V	P2-7	YELLOW	24 AWG
P1-39	AGND	P2-10	ORANGE	24 AWG
P1-40	(+)5V	P2-13	VIOLET	24 AWG
P1-41	RESET	P3-39	BLACK/GRAY	28 AWG
P1-42	DGND	P2-17	GRAY/BLACK	28 AWG
P1-43	SCLK1-	P3-9	BROWN/BLACK	28 AWG
P1-44	SCLK1+	P3-10	BLACK/BROWN	28 AWG
P1-45	SCLK2+	P3-12	RED/BLUE	28 AWG
P1-46	SCLK2-	P3-11	BLUE/RED	28 AWG
P1-47	SUMWELL-	P3-15	RED/GREEN	28 AWG

KEO Consultants

Signal CCA to Camera Head Cable for HAARP

P1-48	SUMWELL+	P3-16	GREEN/RED	28 AWG
P1-49	SCLK3-	P3-13	BLACK/BLUE	28 AWG
P1-50	SCLK3+	P3-14	BLUE/BLACK	28 AWG
P1-51	PCLK1+	P3-21	ORANGE/WHITE	28 AWG
P1-52	PCLK1-	P3-19	WHITE/ORANGE	28 AWG
P1-53	PCLK3+	P3-29	RED/ORANGE	28 AWG
P1-54	PCLK3-	P3-27	ORANGE/RED	28 AWG
P1-55	TGATE+	P3-33	BLUE/YELLOW	28 AWG
P1-56	TGATE-	P3-31	YELLOW/BLUE	28 AWG
P1-57	PCLK2+	P3-25	YELLOW/GREEN	28 AWG
P1-58	PCLK2-	P3-23	GREEN/YELLOW	28 AWG
P1-59	DGND	P2-18	GRAY	24 AWG
P1-60	(+)5V	P2-14	BLUE	24 AWG
P1-61	(+)15V	P2-6	WHITE/RED	24 AWG
P1-62	(-)15V	P2-8	BROWN	24 AWG
P1-63	AGND	P2-12	WHITE	24 AWG

	INTERNAL	CAMERA	CABLE	(CCD Head)
CCD Head	Name	Internal CCA's	Color	Wire Gage
63 PIN Circ.		J3 - Analog CCA		
		J2 - Driver CCA		
J1-1	DAT01	J3-40	ORANGE/YELLOW	28 AWG
J1-2	DGND	J3-41	YELLOW/ORANGE	28 AWG
J1-3	DAT02	J3-37	BLACK/BLUE	28 AWG
J1-4	DGND	J3-38	BLUE/BLACK	28 AWG
J1-5	DAT03	J3-36	ORANGE/BLACK	28 AWG
J1-6	DGND	J3-39	BLACK/ORANGE	28 AWG
J1-7	DAT04	J3-34	RED/BROWN	28 AWG
J1-8	DGND	J3-35	BROWN/RED	28 AWG
J1-9	DAT05	J3-31	WHITE/GREY	28 AWG
J1-10	DGND	J3-32	GREY/WHITE	28 AWG
J1-11	CCDTEMP	J3-6	ORANGE/RED	24 AWG
J1-12	DAT06	J3-30	GREEN/BLACK	28 AWG
J1-13	DGND	J3-33	BLACK/GREEN	28 AWG
J1-14	DAT07	J3-28	RED/ORANGE	28 AWG
J1-15	DGND	J3-29	ORANGE/RED	28 AWG
J1-16	DAT08	J3-25	PURPLE/BLUE	28 AWG
J1-17	DGND	J3-26	BLUE/PURPLE	28 AWG
J1-18	DAT09	J3-24	RED/BLUE	28 AWG
J1-19	DGND	J3-27	BLUE/RED	28 AWG
J1-20	DAT10	J3-22	BLACK/GREY	28 AWG
J1-21	DGND	J3-23	GREY/BLACK	28 AWG
J1-22	DAT11	J3-19	BLUE/WHITE	28 AWG
J1-23	DGND	J3-20	WHITE/BLUE	28 AWG
J1-24	DAT12	J3-18	BLACK/BROWN	28 AWG
J1-25	DGND	J3-21	BROWN/BLACK	28 AWG
J1-26	DCR	J3-16	BROWN/PURPLE	28 AWG
J1-27	GAINSEL2	J3-17	PURPLE/BROWN	28 AWG
J1-28	ADS	J3-42	PURPLE/GREEN	28 AWG
J1-29	GAINSEL2	J3-15	GREEN/PURPLE	28 AWG
J1-30	TEC+	J3-1/J3-4	BROWN/YELLOW	28 AWG
J1-31	TEC-	J3-2/J3-5	BLUE/PURPLE	28 AWG
J1-32	CCDRTD-	J3-11	BROWN/WHITE	24 AWG
J1-33	CCDRTD+	J3-14	WHITE/BROWN	24 AWG
J1-34	TECRTD-	J3-8	YELLOW/GRAY	24 AWG
J1-35	TECRTD+	J3-12	GRAY/YELLOW	24 AWG
J1-36	TECTEMP	J3-9	GREY/PURPLE	24 AWG
J1-37	(+)15V	J3-3	ORANGE	24 AWG
J1-38	(-)15V	J3-7	PURPLE	24 AWG
J1-39	AGND	J3-10	GREEN	24 AWG
J1-40	(+)5V	J3-13	BROWN	24 AWG
				24 AWG
J1-41	RESET	J2-1	BLACK/GRAY	28 AWG
J1-42	DGND	J2-2	GRAY/BLACK	28 AWG
J1-43	SCLK1-	J2-3	GRAY/YELLOW	28 AWG
J1-44	SCLK1+	J2-4	YELLOW/GRAY	28 AWG

KEO Consultants

J1-45	SCLK2+	J2-5	GREEN/RED	28 AWG
J1-46	SCLK2-	J2-6	RED/GREEN	28 AWG
J1-47	SUMWELL-	J2-7	WHITE/ORANGE	28 AWG
J1-48	SUMWELL+	J2-8	ORANGE/WHITE	28 AWG
J1-49	SCLK3-	J2-9	BROWN/YELLOW	28 AWG
J1-50	SCLK3+	J2-10	YELLOW/BROWN	28 AWG
J1-51	PCLK1+	J2-11	WHITE/BROWN	28 AWG
J1-52	PCLK1-	J2-12	BROWN/WHITE	28 AWG
J1-53	PCLK3+	J2-13	BLUE/WHITE	28 AWG
J1-54	PCLK3-	J2-14	WHITE/BLUE	28 AWG
J1-55	TGATE+	J2-15	RED/GRAY	28 AWG
J1-56	TGATE-	J2-16	GRAY/RED	28 AWG
J1-57	PCLK2+	J2-17	GREEN/YELLOW	28 AWG
J1-58	PCLK2-	J2-18	YELLOW/GREEN	28 AWG
J1-59	DGND	J2-24	GREEN	24 AWG
J1-60	(+)5V	J2-23	ORANGE	24 AWG
J1-61	(+)15V	J2-20	YELLOW	24 AWG
J1-62	(-)15V	J2-22	BLUE	24 AWG
J1-63	AGND	J2-21	BROWN	24 AWG

HAARP Shutter Connector Diagram

P1 - Power Conn.

Shutter(+)	Pin 4	Red
Shutter(-)	Pin 1	Black
Switch(+)	Pin 2	Green
Switch(-)	Pin 3	Yellow
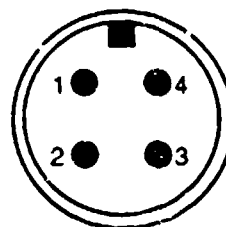
Looking at Socket
LEMO FGG-1B (0.9)

Shutter Coil

Switch

1   4   2   3
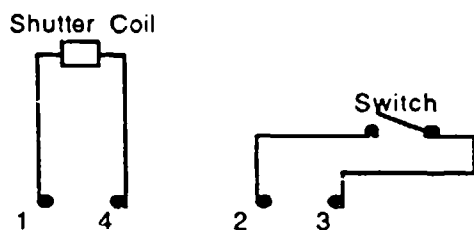
P2 - PhotoDiode Amplifier (Back Shutter Only)

PhotoDiode(+)	Pin 4	Red
Analog Gnd	Pin 1	Black
+15V	Pin 3	Yellow
-15V	Pin 2	Green
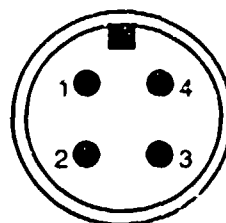
Looking at Socket
LEMO FHG-0B (0.5)
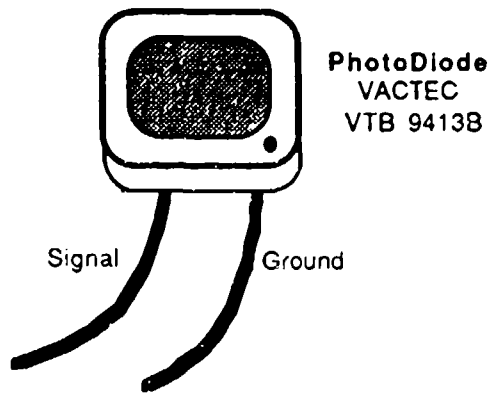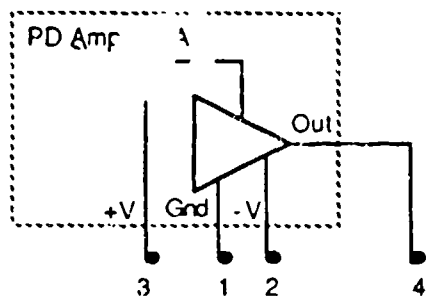
PD Amp

Out

+V   Gnd   -V

3   1   2   4

PhotoDiode
VACTEC
VTB 9413B

Signal   Ground

HAARP DC Wiring Diagram
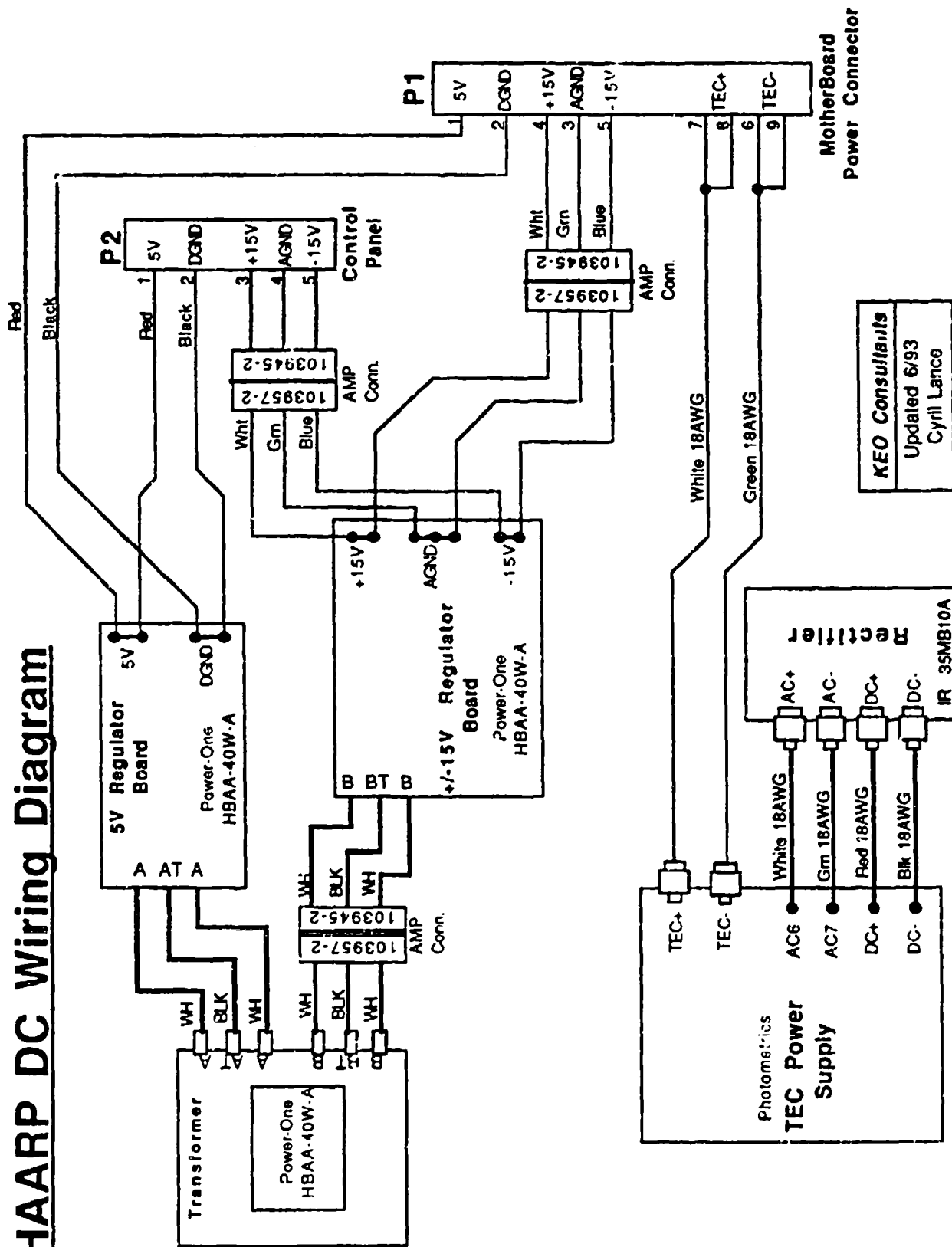
HAARP: Image Intensifierr/TEC Power Cable

Products for Research Intesifier Cooler
Model #: TE342RF
S/N: 18517-92

SIG

P4

Int:P4 4 ——— RED ——— A
Int:P3 3 ——— GREEN ——— B
Int:P2 2 ——— YELLOW ——— C
Int:P1 1 ——— BLACK ——— D

24AWG

Image Intensifier

Int:RTD+ 5
Int:RTD- 6 24AWG Twisted

PWR

RED ——— C
BLACK ——— D
WHITE ——— A
GREEN ——— B

Control Panel

20AWG

TEC+
TEC-

TEC Power Supply

P1

7

WHITE ——— 8

18AWG ——— 6

GREEN ——— 9

MIP Mother Board
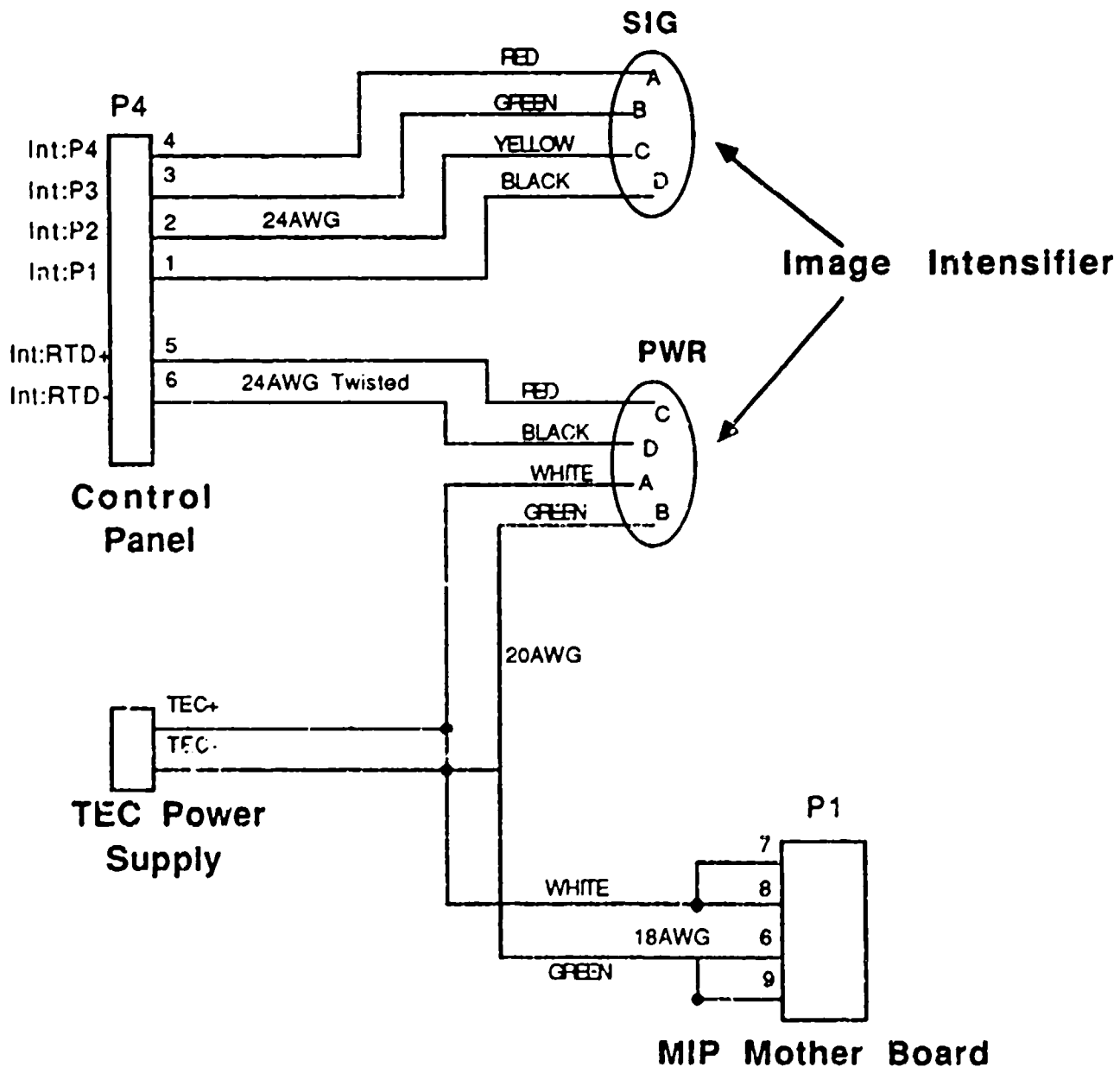
(Note: 4/93 -- Products for Research incorrectly wired the HAARP Intensifier Cooler, so the cabling for the HAARP imager is different than the MIP Imager. This documentation correctly describes the HAARP imager. Unfortunately, the two Intensifier coolers are not directly interchangable as a result of this mistake.)
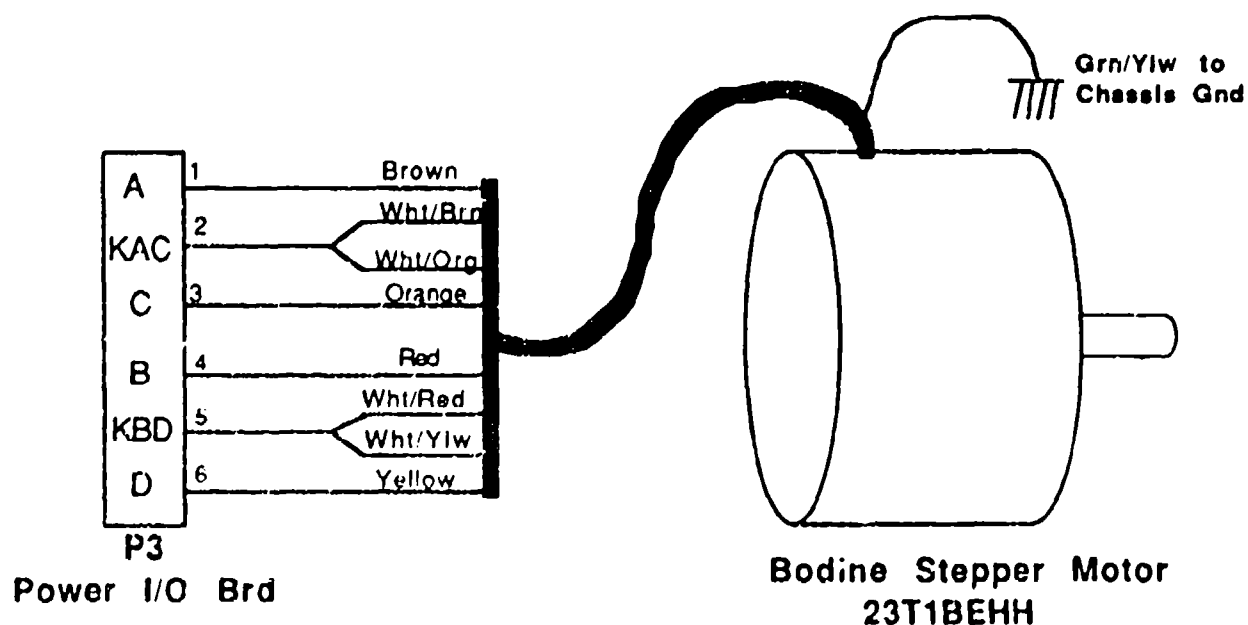
MIP and HAARP Filter Wheel Housing
Connector Diagram

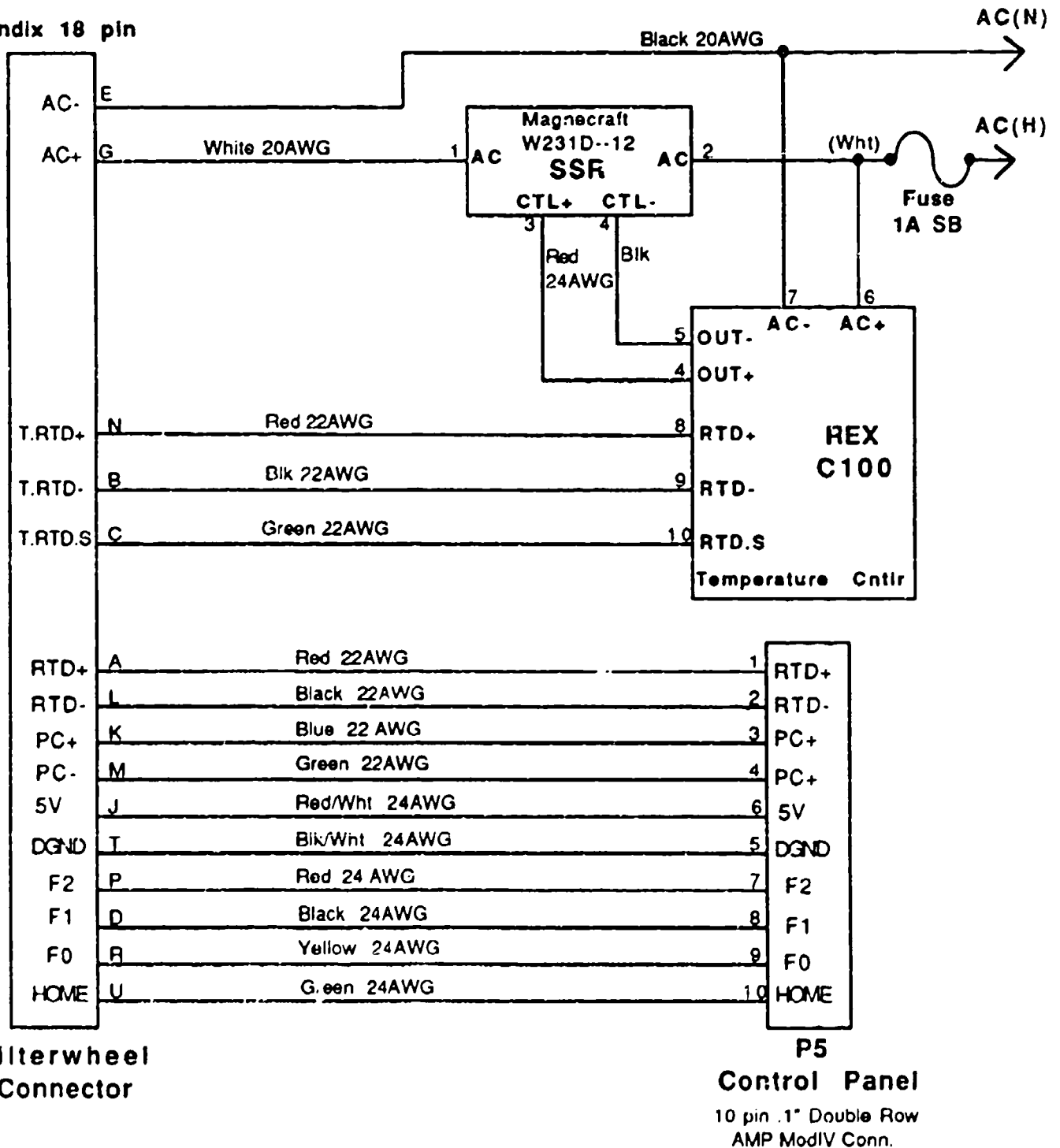|	
AC(+)	G
AC(-)	E
CTL(+)	H
CTL(-)	S
T.RTD(+)	N
T.RTD(-)	B
T.RTD(S)	C
RTD(+)	A
RTD(-)	L
PC(+)	K
PC(-)	M
5V	J
DGND	T
F2	P
F1	D
F0	R
HOME	U
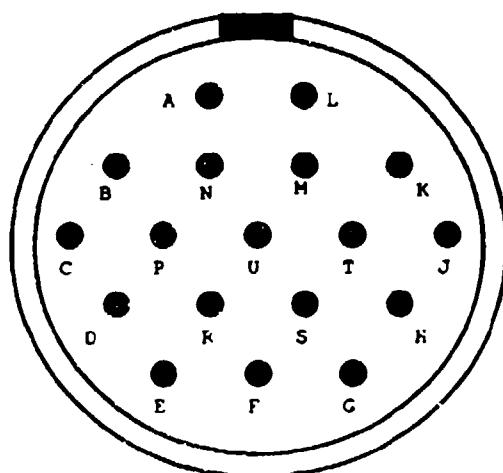
Notes:
- T.RTD connects to REX10 Temperature Controller
- RTD connects to the Amp. to 63HC11
- PC refers to Photoresistive Cell
- CTL refers to SSR control voltage (12V)
- CTL(+) AND CTL(-) [pins H & S] are on the MIP Filterwheel only.

Grn/Ylw to Chassis Gnd

P3 Power I/O Brd		
A	1	Brown
KAC	2	Wht/Brn
		Wht/Org
C	3	Orange
B	4	Red
KBD	5	Wht/Red
		Wht/Ylw
D	6	Yellow

P3
Power I/O Brd

**Bodine Stepper Motor
23T1BEHH**

HAARP Imager
Filter Wheel Cabling

307

11.13 Removing the Image Intensifier from the HAARP Imager

In the event that you need to remove the image intensifier from the HAARP imager, the following steps must be taken:

Unplug the Power to the HAARP imager FIRST!!!

(i) Unplug the Filter Wheel: Disconnect the filter wheel cable inside the imager chassis (18 pin Bendix Connector), and then gently disconnect the Stepper Motor cable from the Power I/O Board that connects to the filterwheel.

(ii) Remove the Filter Wheel: Unsrew the four 8-32 socket heads at the front of the imager chassis (two per side) the hold the filter wheel to the chassis. Gently pull the filter wheel away from the chassis. The re-imaging tube connecting to the front shutter will come off along with the filter wheel (this is normal).

(iii) Remove the front Shutter: From inside the U-channel, remove the LEMO connector attached to the shutter (in between the Power I/O board (front) and the TEC power supply). Remove the three Phillips head 10-32 screws and slide the shutter assembly fowards off of the Cannon 85mm lens. A close-up lens will remain attached to the Cannon lens as the shutter slides off.

(iv) Remove Camera Head: Remove the four 8-32 socket head screws holding the camera to the camera housing. Disconnect the large, black AMP connector on the base of the camera head and slide the head out the back of the housing.

(v) Remove the back Shutter: From inside the U-channel, remove the two LEMO connectors just foward of the circuit card cage that connect to the shutter. Remove the three Phillips head 10-32 screws and slide the shutter assembly backwards.

(vi) Remove the Intensifier housing: From inside the chassis, remove the six 6-32 socket head screws holding the intensifier housing in. These are centered around the main cooling fan (three to a side) but two of the foward screws are tricky to remove. To make the disassembly easier, loosen the REX C10 Temperature Controller and slide it slightly out of the control panel to gain more access to the two screws underneath it.

Once these screws have been removed, the intensifier housing will pull off the chassis, but there are two internal connectors still attached. Gently pull the housing out as far as you can (around 2") and reach in (you might need pliers) and release the two Bendix connectors to the intensifier housing.

(vii) <u>Remove the back cover/relay lens</u>: Unscrew the eleven 4-40 button head screws for the back cover/relay lens unit, and slide the back assembly off of the rest of the housing. The intensifier will now be sticking out of the housing about 1".

(viii) <u>Slide the Intensifier out of housing</u>: Without rotating the tube, pull the tube out of the housing.

Additional parts to the Intensifier cooler:

<u>Intensifier Mask</u>: A thin Delrin Mask inserts into the curvature lens housing directly between the curvature lens and the image intensifier. To remove this mask, complete steps (i) through (viii) above, and with a needle-nose pliers, gently pull the mask out of the curvature lens housing. It is crucial that you remember the correct orientation of this mask with respect to the housing and the cooling unit. The mask and the curvature lens housing are lightly marked with a pencil.

<u>Curvature Lens</u>: To remove the curvature corrector lens from inside the intensifier housing, complete steps (i) through (viii) above, and then remove the four 2-56 nylon screws holding the curvature lens assembly from inside the intensifier housing. This can only be done once the intensifier has been removed. The curvature lens housing should be removed and re-installed with the same orientation with respect to the intensifier cooler, as this will preserve the alignment settings of the instrument.

<u>Cooler Window</u>: To remove the cooler window, complete steps (i) through (iv) above and set the housing down on the bench. Remove the four 6-32 socket head screws from the front cover holding the intensifier cooling unit from inside the housing cover. The cooling unit will now come out of the housing. Remove the four 4-40 screws from the front of the cooling unit which releases the window retaining plate.